



OpenShift Container Platform 4.5

Installing

Installing and configuring OpenShift Container Platform clusters

OpenShift Container Platform 4.5 Installing

Installing and configuring OpenShift Container Platform clusters

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides information about installing OpenShift Container Platform and details about some configuration processes.

Table of Contents

CHAPTER 1. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION	32
1.1. PREREQUISITES	32
1.2. ABOUT THE MIRROR REGISTRY	32
1.3. PREPARING YOUR MIRROR HOST	33
1.3.1. Installing the CLI by downloading the binary	33
1.3.1.1. Installing the CLI on Linux	33
1.3.1.2. Installing the CLI on Windows	33
1.3.1.3. Installing the CLI on macOS	34
1.4. CONFIGURING CREDENTIALS THAT ALLOW IMAGES TO BE MIRRORED	34
1.5. MIRRORING THE OPENSIFT CONTAINER PLATFORM IMAGE REPOSITORY	37
1.6. THE CLUSTER SAMPLES OPERATOR IN A DISCONNECTED ENVIRONMENT	40
1.7. ADDITIONAL RESOURCES	40
CHAPTER 2. INSTALLING ON AWS	41
2.1. CONFIGURING AN AWS ACCOUNT	41
2.1.1. Configuring Route 53	41
2.1.2. AWS account limits	41
2.1.3. Required AWS permissions	43
2.1.4. Creating an IAM user	51
2.1.5. Supported AWS regions	51
2.1.6. Next steps	52
2.2. MANUALLY CREATING IAM FOR AWS	52
2.2.1. Manually create IAM	52
2.2.2. Admin credentials root secret format	54
2.2.2.1. Upgrades	54
2.2.3. Mint Mode	55
2.2.4. Mint Mode with removal or rotation of the admin credential	55
2.3. INSTALLING A CLUSTER QUICKLY ON AWS	55
2.3.1. Prerequisites	55
2.3.2. Internet and Telemetry access for OpenShift Container Platform	56
2.3.3. Generating an SSH private key and adding it to the agent	56
2.3.4. Obtaining the installation program	58
2.3.5. Deploying the cluster	58
2.3.6. Installing the CLI by downloading the binary	60
2.3.6.1. Installing the CLI on Linux	60
2.3.6.2. Installing the CLI on Windows	61
2.3.6.3. Installing the CLI on macOS	61
2.3.7. Logging in to the cluster	62
2.3.8. Next steps	62
2.4. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS	63
2.4.1. Prerequisites	63
2.4.2. Internet and Telemetry access for OpenShift Container Platform	63
2.4.3. Generating an SSH private key and adding it to the agent	64
2.4.4. Obtaining the installation program	65
2.4.5. Creating the installation configuration file	66
2.4.5.1. Installation configuration parameters	67
2.4.5.1.1. Required configuration parameters	67
2.4.5.1.2. Network configuration parameters	69
2.4.5.1.3. Optional configuration parameters	70
2.4.5.1.4. Optional AWS configuration parameters	74
2.4.5.2. Sample customized install-config.yaml file for AWS	75

2.4.6. Deploying the cluster	77
2.4.7. Installing the CLI by downloading the binary	78
2.4.7.1. Installing the CLI on Linux	78
2.4.7.2. Installing the CLI on Windows	79
2.4.7.3. Installing the CLI on macOS	79
2.4.8. Logging in to the cluster	80
2.4.9. Next steps	80
2.5. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS	81
2.5.1. Prerequisites	81
2.5.2. Internet and Telemetry access for OpenShift Container Platform	81
2.5.3. Generating an SSH private key and adding it to the agent	82
2.5.4. Obtaining the installation program	83
2.5.5. Creating the installation configuration file	84
2.5.5.1. Installation configuration parameters	85
2.5.5.1.1. Required configuration parameters	85
2.5.5.1.2. Network configuration parameters	87
2.5.5.1.3. Optional configuration parameters	88
2.5.5.1.4. Optional AWS configuration parameters	92
2.5.5.2. Network configuration parameters	93
2.5.5.3. Sample customized install-config.yaml file for AWS	94
2.5.6. Modifying advanced network configuration parameters	96
2.5.7. Cluster Network Operator configuration	98
2.5.7.1. Configuration parameters for the OpenShift SDN default CNI network provider	99
2.5.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider	99
2.5.7.3. Cluster Network Operator example configuration	100
2.5.8. Deploying the cluster	100
2.5.9. Installing the CLI by downloading the binary	102
2.5.9.1. Installing the CLI on Linux	102
2.5.9.2. Installing the CLI on Windows	102
2.5.9.3. Installing the CLI on macOS	103
2.5.10. Logging in to the cluster	103
2.5.11. Next steps	104
2.6. INSTALLING A CLUSTER ON AWS IN A RESTRICTED NETWORK	104
2.6.1. Prerequisites	104
2.6.2. About installations in restricted networks	105
2.6.2.1. Additional limits	105
2.6.3. About using a custom VPC	105
2.6.3.1. Requirements for using your VPC	106
2.6.3.2. VPC validation	108
2.6.3.3. Division of permissions	108
2.6.3.4. Isolation between clusters	109
2.6.4. Internet and Telemetry access for OpenShift Container Platform	109
2.6.5. Generating an SSH private key and adding it to the agent	109
2.6.6. Creating the installation configuration file	111
2.6.6.1. Installation configuration parameters	113
2.6.6.1.1. Required configuration parameters	113
2.6.6.1.2. Network configuration parameters	115
2.6.6.1.3. Optional configuration parameters	116
2.6.6.1.4. Optional AWS configuration parameters	120
2.6.6.2. Sample customized install-config.yaml file for AWS	121
2.6.6.3. Configuring the cluster-wide proxy during installation	123
2.6.7. Deploying the cluster	125
2.6.8. Installing the CLI by downloading the binary	126

2.6.8.1. Installing the CLI on Linux	126
2.6.8.2. Installing the CLI on Windows	127
2.6.8.3. Installing the CLI on macOS	127
2.6.9. Logging in to the cluster	127
2.6.10. Next steps	128
2.7. INSTALLING A CLUSTER ON AWS INTO AN EXISTING VPC	128
2.7.1. Prerequisites	128
2.7.2. About using a custom VPC	129
2.7.2.1. Requirements for using your VPC	129
2.7.2.2. VPC validation	131
2.7.2.3. Division of permissions	132
2.7.2.4. Isolation between clusters	132
2.7.3. Internet and Telemetry access for OpenShift Container Platform	132
2.7.4. Generating an SSH private key and adding it to the agent	133
2.7.5. Obtaining the installation program	134
2.7.6. Creating the installation configuration file	135
2.7.6.1. Installation configuration parameters	136
2.7.6.1.1. Required configuration parameters	137
2.7.6.1.2. Network configuration parameters	138
2.7.6.1.3. Optional configuration parameters	140
2.7.6.1.4. Optional AWS configuration parameters	143
2.7.6.2. Sample customized install-config.yaml file for AWS	145
2.7.6.3. Configuring the cluster-wide proxy during installation	147
2.7.7. Deploying the cluster	148
2.7.8. Installing the CLI by downloading the binary	150
2.7.8.1. Installing the CLI on Linux	150
2.7.8.2. Installing the CLI on Windows	150
2.7.8.3. Installing the CLI on macOS	151
2.7.9. Logging in to the cluster	151
2.7.10. Next steps	152
2.8. INSTALLING A PRIVATE CLUSTER ON AWS	152
2.8.1. Prerequisites	152
2.8.2. Private clusters	152
2.8.2.1. Private clusters in AWS	153
2.8.2.1.1. Limitations	153
2.8.3. About using a custom VPC	153
2.8.3.1. Requirements for using your VPC	154
2.8.3.2. VPC validation	156
2.8.3.3. Division of permissions	156
2.8.3.4. Isolation between clusters	156
2.8.4. Internet and Telemetry access for OpenShift Container Platform	157
2.8.5. Generating an SSH private key and adding it to the agent	157
2.8.6. Obtaining the installation program	159
2.8.7. Manually creating the installation configuration file	159
2.8.7.1. Installation configuration parameters	160
2.8.7.1.1. Required configuration parameters	160
2.8.7.1.2. Network configuration parameters	162
2.8.7.1.3. Optional configuration parameters	163
2.8.7.1.4. Optional AWS configuration parameters	167
2.8.7.2. Sample customized install-config.yaml file for AWS	168
2.8.7.3. Configuring the cluster-wide proxy during installation	170
2.8.8. Deploying the cluster	172
2.8.9. Installing the CLI by downloading the binary	173

2.8.9.1. Installing the CLI on Linux	173
2.8.9.2. Installing the CLI on Windows	173
2.8.9.3. Installing the CLI on macOS	174
2.8.10. Logging in to the cluster	174
2.8.11. Next steps	175
2.9. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN AWS BY USING CLOUDFORMATION TEMPLATES	175
2.9.1. Prerequisites	175
2.9.2. Internet and Telemetry access for OpenShift Container Platform	176
2.9.3. Required AWS infrastructure components	177
2.9.3.1. Cluster machines	177
2.9.3.2. Certificate signing requests management	178
2.9.3.3. Other infrastructure components	178
2.9.3.4. Required AWS permissions	184
2.9.4. Obtaining the installation program	192
2.9.5. Generating an SSH private key and adding it to the agent	192
2.9.6. Creating the installation files for AWS	194
2.9.6.1. Creating the installation configuration file	194
2.9.6.2. Configuring the cluster-wide proxy during installation	195
2.9.6.3. Creating the Kubernetes manifest and Ignition config files	197
2.9.7. Extracting the infrastructure name	199
2.9.8. Creating a VPC in AWS	199
2.9.8.1. CloudFormation template for the VPC	201
2.9.9. Creating networking and load balancing components in AWS	206
2.9.9.1. CloudFormation template for the network and load balancers	210
2.9.10. Creating security group and roles in AWS	217
2.9.10.1. CloudFormation template for security objects	219
2.9.11. RHCOS AMIs for the AWS infrastructure	228
2.9.12. Creating the bootstrap node in AWS	229
2.9.12.1. CloudFormation template for the bootstrap machine	233
2.9.13. Creating the control plane machines in AWS	237
2.9.13.1. CloudFormation template for control plane machines	242
2.9.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure	249
2.9.14.1. Creating the worker nodes in AWS	250
2.9.14.1.1. CloudFormation template for worker machines	254
2.9.15. Installing the CLI by downloading the binary	256
2.9.15.1. Installing the CLI on Linux	256
2.9.15.2. Installing the CLI on Windows	257
2.9.15.3. Installing the CLI on macOS	257
2.9.16. Logging in to the cluster	258
2.9.17. Approving the certificate signing requests for your machines	258
2.9.18. Initial Operator configuration	260
2.9.18.1. Image registry storage configuration	261
2.9.18.1.1. Configuring registry storage for AWS with user-provisioned infrastructure	262
2.9.18.1.2. Configuring storage for the image registry in non-production clusters	262
2.9.19. Deleting the bootstrap resources	263
2.9.20. Creating the Ingress DNS Records	264
2.9.21. Completing an AWS installation on user-provisioned infrastructure	266
2.9.22. Next steps	267
2.10. INSTALLING A CLUSTER ON AWS IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE	267
2.10.1. Prerequisites	268
2.10.2. About installations in restricted networks	268

2.10.2.1. Additional limits	269
2.10.3. Internet and Telemetry access for OpenShift Container Platform	269
2.10.4. Required AWS infrastructure components	270
2.10.4.1. Cluster machines	270
2.10.4.2. Certificate signing requests management	271
2.10.4.3. Other infrastructure components	271
2.10.4.4. Required AWS permissions	277
2.10.5. Generating an SSH private key and adding it to the agent	285
2.10.6. Creating the installation files for AWS	286
2.10.6.1. Creating the installation configuration file	286
2.10.6.2. Configuring the cluster-wide proxy during installation	288
2.10.6.3. Creating the Kubernetes manifest and Ignition config files	290
2.10.7. Extracting the infrastructure name	292
2.10.8. Creating a VPC in AWS	292
2.10.8.1. CloudFormation template for the VPC	294
2.10.9. Creating networking and load balancing components in AWS	299
2.10.9.1. CloudFormation template for the network and load balancers	303
2.10.10. Creating security group and roles in AWS	310
2.10.10.1. CloudFormation template for security objects	313
2.10.11. RHCOS AMIs for the AWS infrastructure	321
2.10.12. Creating the bootstrap node in AWS	322
2.10.12.1. CloudFormation template for the bootstrap machine	326
2.10.13. Creating the control plane machines in AWS	330
2.10.13.1. CloudFormation template for control plane machines	335
2.10.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure	342
2.10.14.1. Creating the worker nodes in AWS	343
2.10.14.1.1. CloudFormation template for worker machines	347
2.10.15. Logging in to the cluster	349
2.10.16. Approving the certificate signing requests for your machines	350
2.10.17. Initial Operator configuration	352
2.10.17.1. Image registry storage configuration	353
2.10.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure	353
2.10.17.1.2. Configuring storage for the image registry in non-production clusters	354
2.10.18. Deleting the bootstrap resources	355
2.10.19. Creating the Ingress DNS Records	355
2.10.20. Completing an AWS installation on user-provisioned infrastructure	358
2.10.21. Next steps	358
2.11. UNINSTALLING A CLUSTER ON AWS	359
2.11.1. Removing a cluster that uses installer-provisioned infrastructure	359
CHAPTER 3. INSTALLING ON AZURE	360
3.1. CONFIGURING AN AZURE ACCOUNT	360
3.1.1. Azure account limits	360
3.1.2. Configuring a public DNS zone in Azure	362
3.1.3. Increasing Azure account limits	363
3.1.4. Required Azure roles	364
3.1.5. Creating a service principal	364
3.1.6. Supported Azure regions	367
3.1.7. Next steps	368
3.2. MANUALLY CREATING IAM FOR AZURE	368
3.2.1. Manually create IAM	368
3.2.2. Admin credentials root secret format	370
3.2.2.1. Upgrades	371

3.2.3. Mint Mode	371
3.3. INSTALLING A CLUSTER QUICKLY ON AZURE	371
3.3.1. Prerequisites	371
3.3.2. Internet and Telemetry access for OpenShift Container Platform	371
3.3.3. Generating an SSH private key and adding it to the agent	372
3.3.4. Obtaining the installation program	373
3.3.5. Deploying the cluster	374
3.3.6. Installing the CLI by downloading the binary	376
3.3.6.1. Installing the CLI on Linux	376
3.3.6.2. Installing the CLI on Windows	377
3.3.6.3. Installing the CLI on macOS	377
3.3.7. Logging in to the cluster	378
3.3.8. Next steps	378
3.4. INSTALLING A CLUSTER ON AZURE WITH CUSTOMIZATIONS	378
3.4.1. Prerequisites	378
3.4.2. Internet and Telemetry access for OpenShift Container Platform	379
3.4.3. Generating an SSH private key and adding it to the agent	379
3.4.4. Obtaining the installation program	381
3.4.5. Creating the installation configuration file	381
3.4.5.1. Installation configuration parameters	383
3.4.5.1.1. Required configuration parameters	383
3.4.5.1.2. Network configuration parameters	385
3.4.5.1.3. Optional configuration parameters	386
3.4.5.1.4. Additional Azure configuration parameters	390
3.4.5.2. Sample customized install-config.yaml file for Azure	391
3.4.6. Deploying the cluster	393
3.4.7. Installing the CLI by downloading the binary	394
3.4.7.1. Installing the CLI on Linux	394
3.4.7.2. Installing the CLI on Windows	394
3.4.7.3. Installing the CLI on macOS	395
3.4.8. Logging in to the cluster	395
3.4.9. Next steps	396
3.5. INSTALLING A CLUSTER ON AZURE WITH NETWORK CUSTOMIZATIONS	396
3.5.1. Prerequisites	396
3.5.2. Internet and Telemetry access for OpenShift Container Platform	397
3.5.3. Generating an SSH private key and adding it to the agent	397
3.5.4. Obtaining the installation program	398
3.5.5. Creating the installation configuration file	399
3.5.5.1. Installation configuration parameters	401
3.5.5.1.1. Required configuration parameters	401
3.5.5.1.2. Network configuration parameters	402
3.5.5.1.3. Optional configuration parameters	404
3.5.5.1.4. Additional Azure configuration parameters	407
3.5.5.2. Network configuration parameters	408
3.5.5.3. Sample customized install-config.yaml file for Azure	409
3.5.6. Modifying advanced network configuration parameters	411
3.5.7. Cluster Network Operator configuration	413
3.5.7.1. Configuration parameters for the OpenShift SDN default CNI network provider	414
3.5.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider	414
3.5.7.3. Cluster Network Operator example configuration	415
3.5.8. Deploying the cluster	416
3.5.9. Installing the CLI by downloading the binary	417
3.5.9.1. Installing the CLI on Linux	417

3.5.9.2. Installing the CLI on Windows	417
3.5.9.3. Installing the CLI on macOS	418
3.5.10. Logging in to the cluster	418
3.5.11. Next steps	419
3.6. INSTALLING A CLUSTER ON AZURE INTO AN EXISTING VNET	419
3.6.1. Prerequisites	419
3.6.2. About reusing a VNet for your OpenShift Container Platform cluster	419
3.6.2.1. Requirements for using your VNet	419
3.6.2.1.1. Network security group requirements	420
3.6.2.2. Division of permissions	421
3.6.2.3. Isolation between clusters	421
3.6.3. Internet and Telemetry access for OpenShift Container Platform	421
3.6.4. Generating an SSH private key and adding it to the agent	422
3.6.5. Obtaining the installation program	423
3.6.6. Creating the installation configuration file	424
3.6.6.1. Installation configuration parameters	426
3.6.6.1.1. Required configuration parameters	426
3.6.6.1.2. Network configuration parameters	427
3.6.6.1.3. Optional configuration parameters	429
3.6.6.1.4. Additional Azure configuration parameters	433
3.6.6.2. Sample customized install-config.yaml file for Azure	433
3.6.6.3. Configuring the cluster-wide proxy during installation	435
3.6.7. Deploying the cluster	437
3.6.8. Installing the CLI by downloading the binary	438
3.6.8.1. Installing the CLI on Linux	438
3.6.8.2. Installing the CLI on Windows	439
3.6.8.3. Installing the CLI on macOS	439
3.6.9. Logging in to the cluster	440
3.6.10. Next steps	440
3.7. INSTALLING A PRIVATE CLUSTER ON AZURE	440
3.7.1. Prerequisites	440
3.7.2. Private clusters	441
3.7.2.1. Private clusters in Azure	441
3.7.2.1.1. Limitations	441
3.7.3. About reusing a VNet for your OpenShift Container Platform cluster	442
3.7.3.1. Requirements for using your VNet	442
3.7.3.1.1. Network security group requirements	443
3.7.3.2. Division of permissions	443
3.7.3.3. Isolation between clusters	444
3.7.4. Internet and Telemetry access for OpenShift Container Platform	444
3.7.5. Generating an SSH private key and adding it to the agent	444
3.7.6. Obtaining the installation program	446
3.7.7. Manually creating the installation configuration file	446
3.7.7.1. Installation configuration parameters	447
3.7.7.1.1. Required configuration parameters	448
3.7.7.1.2. Network configuration parameters	449
3.7.7.1.3. Optional configuration parameters	450
3.7.7.1.4. Additional Azure configuration parameters	454
3.7.7.2. Sample customized install-config.yaml file for Azure	455
3.7.7.3. Configuring the cluster-wide proxy during installation	457
3.7.8. Deploying the cluster	458
3.7.9. Installing the CLI by downloading the binary	459
3.7.9.1. Installing the CLI on Linux	460

3.7.9.2. Installing the CLI on Windows	460
3.7.9.3. Installing the CLI on macOS	461
3.7.10. Logging in to the cluster	461
3.7.11. Next steps	462
3.8. INSTALLING A CLUSTER ON AZURE USING ARM TEMPLATES	462
3.8.1. Prerequisites	462
3.8.2. Internet and Telemetry access for OpenShift Container Platform	462
3.8.3. Configuring your Azure project	463
3.8.3.1. Azure account limits	463
3.8.3.2. Configuring a public DNS zone in Azure	465
3.8.3.3. Increasing Azure account limits	466
3.8.3.4. Certificate signing requests management	467
3.8.3.5. Required Azure roles	467
3.8.3.6. Creating a service principal	467
3.8.3.7. Supported Azure regions	470
3.8.4. Obtaining the installation program	471
3.8.5. Generating an SSH private key and adding it to the agent	472
3.8.6. Creating the installation files for Azure	473
3.8.6.1. Creating the installation configuration file	473
3.8.6.2. Configuring the cluster-wide proxy during installation	475
3.8.6.3. Exporting common variables for ARM templates	476
3.8.6.4. Creating the Kubernetes manifest and Ignition config files	478
3.8.7. Creating the Azure resource group and identity	480
3.8.8. Uploading the RHCOS cluster image and bootstrap Ignition config file	481
3.8.9. Example for creating DNS zones	482
3.8.10. Creating a VNet in Azure	483
3.8.10.1. ARM template for the VNet	484
3.8.11. Deploying the RHCOS cluster image for the Azure infrastructure	485
3.8.11.1. ARM template for image storage	486
3.8.12. Networking requirements for user-provisioned infrastructure	487
Network topology requirements	488
Load balancers	488
3.8.13. Creating networking and load balancing components in Azure	490
3.8.13.1. ARM template for the network and load balancers	491
3.8.14. Creating the bootstrap machine in Azure	496
3.8.14.1. ARM template for the bootstrap machine	496
3.8.15. Creating the control plane machines in Azure	501
3.8.15.1. ARM template for control plane machines	502
3.8.16. Wait for bootstrap completion and remove bootstrap resources in Azure	508
3.8.17. Creating additional worker machines in Azure	509
3.8.17.1. ARM template for worker machines	510
3.8.18. Installing the CLI by downloading the binary	514
3.8.18.1. Installing the CLI on Linux	514
3.8.18.2. Installing the CLI on Windows	515
3.8.18.3. Installing the CLI on macOS	515
3.8.19. Logging in to the cluster	516
3.8.20. Approving the certificate signing requests for your machines	516
3.8.21. Adding the Ingress DNS records	519
3.8.22. Completing an Azure installation on user-provisioned infrastructure	520
3.9. UNINSTALLING A CLUSTER ON AZURE	521
3.9.1. Removing a cluster that uses installer-provisioned infrastructure	521

CHAPTER 4. INSTALLING ON GCP	522
---	------------

4.1. CONFIGURING A GCP PROJECT	522
4.1.1. Creating a GCP project	522
4.1.2. Enabling API services in GCP	522
4.1.3. Configuring DNS for GCP	523
4.1.4. GCP account limits	524
4.1.5. Creating a service account in GCP	525
4.1.5.1. Required GCP permissions	526
4.1.6. Supported GCP regions	527
4.1.7. Next steps	527
4.2. MANUALLY CREATING IAM FOR GCP	528
4.2.1. Manually create IAM	528
4.2.2. Admin credentials root secret format	529
4.2.2.1. Upgrades	530
4.2.3. Mint Mode	530
4.3. INSTALLING A CLUSTER QUICKLY ON GCP	530
4.3.1. Prerequisites	530
4.3.2. Internet and Telemetry access for OpenShift Container Platform	530
4.3.3. Generating an SSH private key and adding it to the agent	531
4.3.4. Obtaining the installation program	533
4.3.5. Deploying the cluster	533
4.3.6. Installing the CLI by downloading the binary	535
4.3.6.1. Installing the CLI on Linux	536
4.3.6.2. Installing the CLI on Windows	536
4.3.6.3. Installing the CLI on macOS	536
4.3.7. Logging in to the cluster	537
4.3.8. Next steps	537
4.4. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS	538
4.4.1. Prerequisites	538
4.4.2. Internet and Telemetry access for OpenShift Container Platform	538
4.4.3. Generating an SSH private key and adding it to the agent	538
4.4.4. Obtaining the installation program	540
4.4.5. Creating the installation configuration file	541
4.4.5.1. Installation configuration parameters	542
4.4.5.1.1. Required configuration parameters	542
4.4.5.1.2. Network configuration parameters	544
4.4.5.1.3. Optional configuration parameters	545
4.4.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters	549
4.4.5.2. Sample customized install-config.yaml file for GCP	549
4.4.6. Deploying the cluster	551
4.4.7. Installing the CLI by downloading the binary	552
4.4.7.1. Installing the CLI on Linux	553
4.4.7.2. Installing the CLI on Windows	553
4.4.7.3. Installing the CLI on macOS	554
4.4.8. Logging in to the cluster	554
4.4.9. Next steps	555
4.5. INSTALLING A CLUSTER ON GCP WITH NETWORK CUSTOMIZATIONS	555
4.5.1. Prerequisites	555
4.5.2. Internet and Telemetry access for OpenShift Container Platform	555
4.5.3. Generating an SSH private key and adding it to the agent	556
4.5.4. Obtaining the installation program	557
4.5.5. Creating the installation configuration file	558
4.5.5.1. Installation configuration parameters	559
4.5.5.1.1. Required configuration parameters	560

4.5.5.1.2. Network configuration parameters	561
4.5.5.1.3. Optional configuration parameters	563
4.5.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters	566
4.5.5.2. Network configuration parameters	567
4.5.5.3. Sample customized install-config.yaml file for GCP	568
4.5.6. Modifying advanced network configuration parameters	570
4.5.7. Cluster Network Operator configuration	571
4.5.7.1. Configuration parameters for the OpenShift SDN default CNI network provider	572
4.5.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider	573
4.5.7.3. Cluster Network Operator example configuration	573
4.5.8. Deploying the cluster	574
4.5.9. Installing the CLI by downloading the binary	575
4.5.9.1. Installing the CLI on Linux	575
4.5.9.2. Installing the CLI on Windows	576
4.5.9.3. Installing the CLI on macOS	576
4.5.10. Logging in to the cluster	577
4.5.11. Next steps	577
4.6. INSTALLING A CLUSTER ON GCP IN A RESTRICTED NETWORK	577
4.6.1. Prerequisites	577
4.6.2. About installations in restricted networks	578
4.6.2.1. Additional limits	578
4.6.3. Internet and Telemetry access for OpenShift Container Platform	578
4.6.4. Generating an SSH private key and adding it to the agent	579
4.6.5. Creating the installation configuration file	581
4.6.5.1. Installation configuration parameters	583
4.6.5.1.1. Required configuration parameters	583
4.6.5.1.2. Network configuration parameters	585
4.6.5.1.3. Optional configuration parameters	586
4.6.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters	590
4.6.5.2. Sample customized install-config.yaml file for GCP	590
4.6.5.3. Configuring the cluster-wide proxy during installation	593
4.6.6. Deploying the cluster	594
4.6.7. Installing the CLI by downloading the binary	596
4.6.7.1. Installing the CLI on Linux	596
4.6.7.2. Installing the CLI on Windows	596
4.6.7.3. Installing the CLI on macOS	597
4.6.8. Logging in to the cluster	597
4.6.9. Next steps	598
4.7. INSTALLING A CLUSTER ON GCP INTO AN EXISTING VPC	598
4.7.1. Prerequisites	598
4.7.2. Internet and Telemetry access for OpenShift Container Platform	598
4.7.3. Generating an SSH private key and adding it to the agent	599
4.7.4. Obtaining the installation program	600
4.7.5. Creating the installation configuration file	601
4.7.5.1. Installation configuration parameters	602
4.7.5.1.1. Required configuration parameters	603
4.7.5.1.2. Network configuration parameters	604
4.7.5.1.3. Optional configuration parameters	606
4.7.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters	610
4.7.5.2. Sample customized install-config.yaml file for GCP	610
4.7.5.3. Configuring the cluster-wide proxy during installation	612
4.7.6. Deploying the cluster	613
4.7.7. Installing the CLI by downloading the binary	615

4.7.7.1. Installing the CLI on Linux	615
4.7.7.2. Installing the CLI on Windows	615
4.7.7.3. Installing the CLI on macOS	616
4.7.8. Logging in to the cluster	616
4.7.9. Next steps	617
4.8. INSTALLING A PRIVATE CLUSTER ON GCP	617
4.8.1. Prerequisites	617
4.8.2. Private clusters	617
4.8.2.1. Private clusters in GCP	618
4.8.2.1.1. Limitations	618
4.8.3. About using a custom VPC	619
4.8.3.1. Requirements for using your VPC	619
4.8.3.2. Division of permissions	619
4.8.3.3. Isolation between clusters	620
4.8.4. Internet and Telemetry access for OpenShift Container Platform	620
4.8.5. Generating an SSH private key and adding it to the agent	621
4.8.6. Obtaining the installation program	622
4.8.7. Manually creating the installation configuration file	623
4.8.7.1. Installation configuration parameters	624
4.8.7.1.1. Required configuration parameters	624
4.8.7.1.2. Network configuration parameters	625
4.8.7.1.3. Optional configuration parameters	627
4.8.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters	630
4.8.7.2. Sample customized install-config.yaml file for GCP	631
4.8.7.3. Configuring the cluster-wide proxy during installation	633
4.8.8. Deploying the cluster	634
4.8.9. Installing the CLI by downloading the binary	635
4.8.9.1. Installing the CLI on Linux	635
4.8.9.2. Installing the CLI on Windows	636
4.8.9.3. Installing the CLI on macOS	636
4.8.10. Logging in to the cluster	637
4.8.11. Next steps	637
4.9. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES	638
4.9.1. Prerequisites	638
4.9.2. Certificate signing requests management	638
4.9.3. Configuring your GCP project	638
4.9.3.1. Creating a GCP project	638
4.9.3.2. Enabling API services in GCP	639
4.9.3.3. Configuring DNS for GCP	640
4.9.3.4. GCP account limits	640
4.9.3.5. Creating a service account in GCP	642
4.9.3.5.1. Required GCP permissions	642
4.9.3.6. Supported GCP regions	643
4.9.3.7. Installing and configuring CLI tools for GCP	644
4.9.4. Creating the installation files for GCP	645
4.9.4.1. Creating the installation configuration file	645
4.9.4.2. Configuring the cluster-wide proxy during installation	646
4.9.4.3. Creating the Kubernetes manifest and Ignition config files	648
4.9.5. Exporting common variables	650
4.9.5.1. Extracting the infrastructure name	650
4.9.5.2. Exporting common variables for Deployment Manager templates	651
4.9.6. Creating a VPC in GCP	651

4.9.6.1. Deployment Manager template for the VPC	652
4.9.7. Networking requirements for user-provisioned infrastructure	653
Network topology requirements	654
Load balancers	655
4.9.8. Creating load balancers in GCP	656
4.9.8.1. Deployment Manager template for the external load balancer	658
4.9.8.2. Deployment Manager template for the internal load balancer	659
4.9.9. Creating a private DNS zone in GCP	660
4.9.9.1. Deployment Manager template for the private DNS	662
4.9.10. Creating firewall rules in GCP	662
4.9.10.1. Deployment Manager template for firewall rules	663
4.9.11. Creating IAM roles in GCP	666
4.9.11.1. Deployment Manager template for IAM roles	667
4.9.12. Creating the RHCOS cluster image for the GCP infrastructure	668
4.9.13. Creating the bootstrap machine in GCP	669
4.9.13.1. Deployment Manager template for the bootstrap machine	671
4.9.14. Creating the control plane machines in GCP	672
4.9.14.1. Deployment Manager template for control plane machines	674
4.9.15. Wait for bootstrap completion and remove bootstrap resources in GCP	676
4.9.16. Creating additional worker machines in GCP	677
4.9.16.1. Deployment Manager template for worker machines	679
4.9.17. Installing the CLI by downloading the binary	680
4.9.17.1. Installing the CLI on Linux	680
4.9.17.2. Installing the CLI on Windows	681
4.9.17.3. Installing the CLI on macOS	681
4.9.18. Logging in to the cluster	682
4.9.19. Approving the certificate signing requests for your machines	682
4.9.20. Optional: Adding the ingress DNS records	684
4.9.21. Completing a GCP installation on user-provisioned infrastructure	686
4.9.22. Next steps	688
4.10. INSTALLING A CLUSTER WITH SHARED VPC ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES	689
4.10.1. Prerequisites	689
4.10.2. Certificate signing requests management	689
4.10.3. Configuring the GCP project that hosts your cluster	689
4.10.3.1. Creating a GCP project	690
4.10.3.2. Enabling API services in GCP	690
4.10.3.3. GCP account limits	691
4.10.3.4. Creating a service account in GCP	692
4.10.3.4.1. Required GCP permissions	693
4.10.3.5. Supported GCP regions	694
4.10.3.6. Installing and configuring CLI tools for GCP	695
4.10.4. Configuring the GCP project that hosts your shared VPC network	695
4.10.4.1. Configuring DNS for GCP	696
4.10.4.2. Creating a VPC in GCP	697
4.10.4.2.1. Deployment Manager template for the VPC	698
4.10.5. Creating the installation files for GCP	699
4.10.5.1. Manually creating the installation configuration file	700
4.10.5.2. Sample customized install-config.yaml file for GCP	700
4.10.5.3. Configuring the cluster-wide proxy during installation	702
4.10.5.4. Creating the Kubernetes manifest and Ignition config files	704
4.10.6. Exporting common variables	706
4.10.6.1. Extracting the infrastructure name	707

4.10.6.2. Exporting common variables for Deployment Manager templates	707
4.10.7. Networking requirements for user-provisioned infrastructure	708
Network topology requirements	709
Load balancers	709
4.10.8. Creating load balancers in GCP	711
4.10.8.1. Deployment Manager template for the external load balancer	713
4.10.8.2. Deployment Manager template for the internal load balancer	714
4.10.9. Creating a private DNS zone in GCP	715
4.10.9.1. Deployment Manager template for the private DNS	717
4.10.10. Creating firewall rules in GCP	717
4.10.10.1. Deployment Manager template for firewall rules	718
4.10.11. Creating IAM roles in GCP	721
4.10.11.1. Deployment Manager template for IAM roles	723
4.10.12. Creating the RHCOS cluster image for the GCP infrastructure	723
4.10.13. Creating the bootstrap machine in GCP	724
4.10.13.1. Deployment Manager template for the bootstrap machine	726
4.10.14. Creating the control plane machines in GCP	727
4.10.14.1. Deployment Manager template for control plane machines	729
4.10.15. Wait for bootstrap completion and remove bootstrap resources in GCP	732
4.10.16. Creating additional worker machines in GCP	732
4.10.16.1. Deployment Manager template for worker machines	734
4.10.17. Installing the CLI by downloading the binary	735
4.10.17.1. Installing the CLI on Linux	735
4.10.17.2. Installing the CLI on Windows	736
4.10.17.3. Installing the CLI on macOS	736
4.10.18. Logging in to the cluster	737
4.10.19. Approving the certificate signing requests for your machines	737
4.10.20. Adding the ingress DNS records	740
4.10.21. Adding ingress firewall rules	742
4.10.21.1. Creating cluster-wide firewall rules for a shared VPC in GCP	742
4.10.22. Completing a GCP installation on user-provisioned infrastructure	743
4.10.23. Next steps	745
4.11. INSTALLING A CLUSTER ON GCP IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE	746
4.11.1. Prerequisites	746
4.11.2. Configuring your GCP project	746
4.11.2.1. Creating a GCP project	746
4.11.2.2. Enabling API services in GCP	747
4.11.2.3. Configuring DNS for GCP	748
4.11.2.4. GCP account limits	748
4.11.2.5. Creating a service account in GCP	750
4.11.2.5.1. Required GCP permissions	750
4.11.2.6. Supported GCP regions	751
4.11.2.7. Installing and configuring CLI tools for GCP	752
4.11.3. Creating the installation files for GCP	753
4.11.3.1. Creating the installation configuration file	753
4.11.3.2. Creating the Kubernetes manifest and Ignition config files	754
4.11.4. Exporting common variables	756
4.11.4.1. Extracting the infrastructure name	756
4.11.4.2. Exporting common variables for Deployment Manager templates	757
4.11.5. Creating a VPC in GCP	757
4.11.5.1. Deployment Manager template for the VPC	758
4.11.6. Networking requirements for user-provisioned infrastructure	760

Network topology requirements	760
Load balancers	760
4.11.7. Creating load balancers in GCP	762
4.11.7.1. Deployment Manager template for the external load balancer	764
4.11.7.2. Deployment Manager template for the internal load balancer	765
4.11.8. Creating a private DNS zone in GCP	766
4.11.8.1. Deployment Manager template for the private DNS	768
4.11.9. Creating firewall rules in GCP	768
4.11.9.1. Deployment Manager template for firewall rules	769
4.11.10. Creating IAM roles in GCP	772
4.11.10.1. Deployment Manager template for IAM roles	773
4.11.11. Creating the RHCOS cluster image for the GCP infrastructure	774
4.11.12. Creating the bootstrap machine in GCP	775
4.11.12.1. Deployment Manager template for the bootstrap machine	777
4.11.13. Creating the control plane machines in GCP	778
4.11.13.1. Deployment Manager template for control plane machines	780
4.11.14. Wait for bootstrap completion and remove bootstrap resources in GCP	782
4.11.15. Creating additional worker machines in GCP	783
4.11.15.1. Deployment Manager template for worker machines	785
4.11.16. Logging in to the cluster	786
4.11.17. Approving the certificate signing requests for your machines	787
4.11.18. Optional: Adding the ingress DNS records	789
4.11.19. Completing a GCP installation on user-provisioned infrastructure	790
4.11.20. Next steps	793
4.12. UNINSTALLING A CLUSTER ON GCP	793
4.12.1. Removing a cluster that uses installer-provisioned infrastructure	793
CHAPTER 5. INSTALLING ON BARE METAL	795
5.1. INSTALLING A CLUSTER ON BARE METAL	795
5.1.1. Prerequisites	795
5.1.2. Internet and Telemetry access for OpenShift Container Platform	795
5.1.3. Machine requirements for a cluster with user-provisioned infrastructure	796
5.1.3.1. Required machines	796
5.1.3.2. Network connectivity requirements	796
5.1.3.3. Minimum resource requirements	796
5.1.3.4. Certificate signing requests management	797
5.1.4. Creating the user-provisioned infrastructure	797
5.1.4.1. Networking requirements for user-provisioned infrastructure	797
Network topology requirements	798
Load balancers	799
5.1.4.2. User-provisioned DNS requirements	800
5.1.5. Generating an SSH private key and adding it to the agent	803
5.1.6. Obtaining the installation program	804
5.1.7. Installing the CLI by downloading the binary	805
5.1.7.1. Installing the CLI on Linux	805
5.1.7.2. Installing the CLI on Windows	806
5.1.7.3. Installing the CLI on macOS	806
5.1.8. Manually creating the installation configuration file	807
5.1.8.1. Installation configuration parameters	807
5.1.8.1.1. Required configuration parameters	808
5.1.8.1.2. Network configuration parameters	809
5.1.8.1.3. Optional configuration parameters	811
5.1.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters	815

5.1.8.2. Sample install-config.yaml file for bare metal	815
5.1.8.3. Configuring the cluster-wide proxy during installation	817
5.1.9. Configuring a three-node cluster	819
5.1.10. Creating the Kubernetes manifest and Ignition config files	819
5.1.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	821
5.1.11.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	821
5.1.11.1.1. Configure advanced networking	823
5.1.11.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting	824
5.1.12. Creating the cluster	828
5.1.13. Logging in to the cluster	828
5.1.14. Approving the certificate signing requests for your machines	829
5.1.15. Initial Operator configuration	831
5.1.15.1. Image registry removed during installation	832
5.1.15.2. Image registry storage configuration	832
5.1.15.2.1. Configuring registry storage for bare metal	833
5.1.15.2.2. Configuring storage for the image registry in non-production clusters	834
5.1.15.2.3. Configuring block registry storage for bare metal	835
5.1.16. Completing installation on user-provisioned infrastructure	835
5.1.17. Next steps	837
5.2. INSTALLING A CLUSTER ON BARE METAL WITH NETWORK CUSTOMIZATIONS	837
5.2.1. Prerequisites	837
5.2.2. Internet and Telemetry access for OpenShift Container Platform	838
5.2.3. Machine requirements for a cluster with user-provisioned infrastructure	838
5.2.3.1. Required machines	838
5.2.3.2. Network connectivity requirements	839
5.2.3.3. Minimum resource requirements	839
5.2.3.4. Certificate signing requests management	839
5.2.4. Creating the user-provisioned infrastructure	839
5.2.4.1. Networking requirements for user-provisioned infrastructure	840
Network topology requirements	841
Load balancers	841
5.2.4.2. User-provisioned DNS requirements	843
5.2.5. Generating an SSH private key and adding it to the agent	845
5.2.6. Obtaining the installation program	847
5.2.7. Installing the CLI by downloading the binary	848
5.2.7.1. Installing the CLI on Linux	848
5.2.7.2. Installing the CLI on Windows	848
5.2.7.3. Installing the CLI on macOS	849
5.2.8. Manually creating the installation configuration file	849
5.2.8.1. Installation configuration parameters	850
5.2.8.1.1. Required configuration parameters	850
5.2.8.1.2. Network configuration parameters	852
5.2.8.1.3. Optional configuration parameters	853
5.2.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters	857
5.2.8.2. Sample install-config.yaml file for bare metal	857
5.2.8.3. Network configuration parameters	859
5.2.9. Modifying advanced network configuration parameters	860
5.2.10. Cluster Network Operator configuration	862
5.2.10.1. Configuration parameters for the OpenShift SDN default CNI network provider	863
5.2.10.2. Cluster Network Operator example configuration	863
5.2.11. Creating the Ignition config files	864
5.2.12. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	865
5.2.12.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	865

5.2.12.1.1. Configure advanced networking	867
5.2.12.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting	868
5.2.13. Creating the cluster	872
5.2.14. Logging in to the cluster	872
5.2.15. Approving the certificate signing requests for your machines	873
5.2.16. Initial Operator configuration	875
5.2.16.1. Image registry removed during installation	876
5.2.16.2. Image registry storage configuration	876
5.2.16.3. Configuring block registry storage for bare metal	877
5.2.17. Completing installation on user-provisioned infrastructure	877
5.2.18. Next steps	879
5.3. INSTALLING A CLUSTER ON BARE METAL IN A RESTRICTED NETWORK	880
5.3.1. Prerequisites	880
5.3.2. About installations in restricted networks	880
5.3.2.1. Additional limits	881
5.3.3. Internet and Telemetry access for OpenShift Container Platform	881
5.3.4. Machine requirements for a cluster with user-provisioned infrastructure	881
5.3.4.1. Required machines	881
5.3.4.2. Network connectivity requirements	882
5.3.4.3. Minimum resource requirements	882
5.3.4.4. Certificate signing requests management	883
5.3.5. Creating the user-provisioned infrastructure	883
5.3.5.1. Networking requirements for user-provisioned infrastructure	883
Network topology requirements	884
Load balancers	884
5.3.5.2. User-provisioned DNS requirements	886
5.3.6. Generating an SSH private key and adding it to the agent	888
5.3.7. Manually creating the installation configuration file	890
5.3.7.1. Installation configuration parameters	891
5.3.7.1.1. Required configuration parameters	891
5.3.7.1.2. Network configuration parameters	893
5.3.7.1.3. Optional configuration parameters	894
5.3.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters	898
5.3.7.2. Sample install-config.yaml file for bare metal	898
5.3.7.3. Configuring the cluster-wide proxy during installation	901
5.3.8. Configuring a three-node cluster	902
5.3.9. Creating the Kubernetes manifest and Ignition config files	902
5.3.10. Configuring chrony time service	905
5.3.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	906
5.3.11.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	906
5.3.11.1.1. Configure advanced networking	908
5.3.11.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting	909
5.3.12. Creating the cluster	913
5.3.13. Logging in to the cluster	913
5.3.14. Approving the certificate signing requests for your machines	914
5.3.15. Initial Operator configuration	916
5.3.15.1. Image registry storage configuration	917
5.3.15.1.1. Changing the image registry's management state	917
5.3.15.1.2. Configuring registry storage for bare metal	918
5.3.15.1.3. Configuring storage for the image registry in non-production clusters	919
5.3.15.1.4. Configuring block registry storage for bare metal	919
5.3.16. Completing installation on user-provisioned infrastructure	920
5.3.17. Next steps	922

CHAPTER 6. INSTALLING ON IBM Z AND LINUXONE	923
6.1. INSTALLING A CLUSTER ON IBM Z AND LINUXONE	923
6.1.1. Prerequisites	923
6.1.2. Internet and Telemetry access for OpenShift Container Platform	923
6.1.3. Machine requirements for a cluster with user-provisioned infrastructure	924
6.1.3.1. Required machines	924
6.1.3.2. Network connectivity requirements	924
6.1.3.3. IBM Z network connectivity requirements	924
6.1.3.4. Minimum resource requirements	925
6.1.3.5. Minimum IBM Z system requirements	925
Hardware requirements	925
Operating system requirements	925
Disk storage for the z/VM guest virtual machines	925
Storage / Main Memory	926
6.1.3.6. Preferred IBM Z system requirements	926
Hardware requirements	926
Operating system requirements	926
Disk storage for the z/VM guest virtual machines	926
Storage / Main Memory	926
6.1.3.7. Certificate signing requests management	926
6.1.4. Creating the user-provisioned infrastructure	927
6.1.4.1. Networking requirements for user-provisioned infrastructure	927
Network topology requirements	928
Load balancers	928
6.1.4.2. User-provisioned DNS requirements	930
6.1.5. Generating an SSH private key and adding it to the agent	933
6.1.6. Obtaining the installation program	934
6.1.7. Installing the CLI by downloading the binary	935
6.1.7.1. Installing the CLI on Linux	935
6.1.7.2. Installing the CLI on Windows	936
6.1.7.3. Installing the CLI on macOS	936
6.1.8. Manually creating the installation configuration file	937
6.1.8.1. Sample install-config.yaml file for IBM Z	937
6.1.9. Creating the Kubernetes manifest and Ignition config files	939
6.1.10. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	941
6.1.11. Creating the cluster	943
6.1.12. Logging in to the cluster	944
6.1.13. Approving the certificate signing requests for your machines	945
6.1.14. Initial Operator configuration	947
6.1.14.1. Image registry storage configuration	948
6.1.14.1.1. Configuring registry storage for bare metal	948
6.1.14.1.2. Configuring storage for the image registry in non-production clusters	949
6.1.15. Completing installation on user-provisioned infrastructure	950
6.1.16. Collecting debugging information	952
6.1.17. Additional resources	953
6.1.18. Next steps	953
6.2. INSTALLING A CLUSTER ON IBM Z AND LINUXONE IN A RESTRICTED NETWORK	953
6.2.1. About installations in restricted networks	954
6.2.1.1. Additional limits	954
6.2.2. Machine requirements for a cluster with user-provisioned infrastructure	954
6.2.2.1. Required machines	954
6.2.2.2. Network connectivity requirements	955
6.2.2.3. IBM Z network connectivity requirements	955

6.2.2.4. Minimum resource requirements	955
6.2.2.5. Minimum IBM Z system requirements	955
Hardware requirements	955
Operating system requirements	956
Disk storage for the z/VM guest virtual machines	956
Storage / Main Memory	956
6.2.2.6. Preferred IBM Z system requirements	956
Hardware requirements	956
Operating system requirements	956
Disk storage for the z/VM guest virtual machines	957
Storage / Main Memory	957
6.2.2.7. Certificate signing requests management	957
6.2.3. Creating the user-provisioned infrastructure	957
6.2.3.1. Networking requirements for user-provisioned infrastructure	958
Network topology requirements	958
Load balancers	959
6.2.3.2. User-provisioned DNS requirements	960
6.2.4. Generating an SSH private key and adding it to the agent	963
6.2.5. Manually creating the installation configuration file	964
6.2.5.1. Sample install-config.yaml file for IBM Z	965
6.2.5.2. Configuring the cluster-wide proxy during installation	967
6.2.6. Creating the Kubernetes manifest and Ignition config files	969
6.2.7. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	971
6.2.8. Creating the cluster	972
6.2.9. Logging in to the cluster	973
6.2.10. Approving the certificate signing requests for your machines	974
6.2.11. Initial Operator configuration	976
6.2.11.1. Image registry storage configuration	977
6.2.11.1.1. Configuring registry storage for bare metal	977
6.2.11.1.2. Configuring storage for the image registry in non-production clusters	978
6.2.12. Completing installation on user-provisioned infrastructure	979
6.2.13. Collecting debugging information	981
CHAPTER 7. INSTALLING ON IBM POWER	983
7.1. INSTALLING A CLUSTER ON IBM POWER	983
7.1.1. Internet and Telemetry access for OpenShift Container Platform	983
7.1.2. Machine requirements for a cluster with user-provisioned infrastructure	984
7.1.2.1. Required machines	984
7.1.2.2. Network connectivity requirements	984
7.1.2.3. Minimum resource requirements	984
7.1.2.4. Certificate signing requests management	985
7.1.3. Creating the user-provisioned infrastructure	985
7.1.3.1. Networking requirements for user-provisioned infrastructure	985
Network topology requirements	986
Load balancers	986
7.1.3.2. User-provisioned DNS requirements	988
7.1.4. Generating an SSH private key and adding it to the agent	991
7.1.5. Obtaining the installation program	992
7.1.6. Installing the CLI by downloading the binary	993
7.1.6.1. Installing the CLI on Linux	993
7.1.6.2. Installing the CLI on Windows	994
7.1.6.3. Installing the CLI on macOS	994
7.1.7. Manually creating the installation configuration file	995

7.1.7.1. Sample install-config.yaml file for IBM Power	995
7.1.8. Creating the Kubernetes manifest and Ignition config files	997
7.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	999
7.1.9.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	999
7.1.9.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE booting	1001
7.1.10. Creating the cluster	1004
7.1.11. Logging in to the cluster	1005
7.1.12. Approving the certificate signing requests for your machines	1005
7.1.13. Initial Operator configuration	1007
7.1.13.1. Image registry storage configuration	1008
7.1.13.1.1. Configuring registry storage for bare metal	1009
7.1.13.1.2. Configuring storage for the image registry in non-production clusters	1010
7.1.14. Completing installation on user-provisioned infrastructure	1010
7.2. INSTALLING A CLUSTER ON IBM POWER IN A RESTRICTED NETWORK	1013
7.2.1. About installations in restricted networks	1013
7.2.1.1. Additional limits	1014
7.2.2. Machine requirements for a cluster with user-provisioned infrastructure	1014
7.2.2.1. Required machines	1014
7.2.2.2. Network connectivity requirements	1014
7.2.2.3. Minimum resource requirements	1015
7.2.2.4. Certificate signing requests management	1015
7.2.3. Creating the user-provisioned infrastructure	1015
7.2.3.1. Networking requirements for user-provisioned infrastructure	1016
Network topology requirements	1016
Load balancers	1017
7.2.3.2. User-provisioned DNS requirements	1018
7.2.4. Generating an SSH private key and adding it to the agent	1021
7.2.5. Manually creating the installation configuration file	1022
7.2.5.1. Sample install-config.yaml file for IBM Power	1023
7.2.5.2. Configuring the cluster-wide proxy during installation	1025
7.2.6. Creating the Kubernetes manifest and Ignition config files	1027
7.2.7. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines	1029
7.2.7.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image	1029
7.2.7.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE booting	1031
7.2.8. Creating the cluster	1033
7.2.9. Logging in to the cluster	1034
7.2.10. Approving the certificate signing requests for your machines	1035
7.2.11. Initial Operator configuration	1037
7.2.11.1. Image registry storage configuration	1038
7.2.11.1.1. Configuring registry storage for bare metal	1038
7.2.11.1.2. Configuring storage for the image registry in non-production clusters	1039
7.2.12. Completing installation on user-provisioned infrastructure	1040
CHAPTER 8. INSTALLING ON OPENSTACK	1043
8.1. INSTALLING A CLUSTER ON OPENSTACK WITH CUSTOMIZATIONS	1043
8.1.1. Prerequisites	1043
8.1.2. Resource guidelines for installing OpenShift Container Platform on RHOSP	1043
8.1.2.1. Control plane and compute machines	1044
8.1.2.2. Bootstrap machine	1044
8.1.3. Internet and Telemetry access for OpenShift Container Platform	1045
8.1.4. Enabling Swift on RHOSP	1045
8.1.5. Verifying external network access	1046
8.1.6. Defining parameters for the installation program	1047

8.1.7. Obtaining the installation program	1049
8.1.8. Creating the installation configuration file	1049
8.1.8.1. Configuring the cluster-wide proxy during installation	1051
8.1.9. Installation configuration parameters	1052
8.1.9.1. Required configuration parameters	1053
8.1.9.2. Network configuration parameters	1054
8.1.9.3. Optional configuration parameters	1056
8.1.9.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters	1059
8.1.9.5. Optional RHOSP configuration parameters	1060
8.1.9.6. Additional Google Cloud Platform (GCP) configuration parameters	1062
8.1.9.7. Custom subnets in RHOSP deployments	1062
8.1.9.8. Sample customized install-config.yaml file for RHOSP	1063
8.1.10. Generating an SSH private key and adding it to the agent	1064
8.1.11. Enabling access to the environment	1065
8.1.11.1. Enabling access with floating IP addresses	1065
8.1.11.2. Enabling access without floating IP addresses	1066
8.1.12. Deploying the cluster	1066
8.1.13. Verifying cluster status	1067
8.1.14. Logging in to the cluster	1068
8.1.15. Configuring application access with floating IP addresses	1069
8.1.16. Next steps	1069
8.2. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR	1070
8.2.1. Prerequisites	1070
8.2.2. About Kuryr SDN	1070
8.2.3. Resource guidelines for installing OpenShift Container Platform on RHOSP with Kuryr	1071
8.2.3.1. Increasing quota	1073
8.2.3.2. Configuring Neutron	1073
8.2.3.3. Configuring Octavia	1073
8.2.3.3.1. The Octavia OVN Driver	1076
8.2.3.4. Known limitations of installing with Kuryr	1077
RHOSP general limitations	1077
RHOSP version limitations	1077
RHOSP environment limitations	1078
RHOSP upgrade limitations	1078
8.2.3.5. Control plane and compute machines	1078
8.2.3.6. Bootstrap machine	1079
8.2.4. Internet and Telemetry access for OpenShift Container Platform	1079
8.2.5. Enabling Swift on RHOSP	1080
8.2.6. Verifying external network access	1080
8.2.7. Defining parameters for the installation program	1081
8.2.8. Obtaining the installation program	1083
8.2.9. Creating the installation configuration file	1084
8.2.9.1. Configuring the cluster-wide proxy during installation	1085
8.2.10. Installation configuration parameters	1087
8.2.10.1. Required configuration parameters	1087
8.2.10.2. Network configuration parameters	1089
8.2.10.3. Optional configuration parameters	1090
8.2.10.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters	1094
8.2.10.5. Optional RHOSP configuration parameters	1094
8.2.10.6. Additional Google Cloud Platform (GCP) configuration parameters	1096
8.2.10.7. Custom subnets in RHOSP deployments	1096
8.2.10.8. Sample customized install-config.yaml file for RHOSP with Kuryr	1097
8.2.11. Generating an SSH private key and adding it to the agent	1098

8.2.12. Enabling access to the environment	1099
8.2.12.1. Enabling access with floating IP addresses	1100
8.2.12.2. Enabling access without floating IP addresses	1100
8.2.13. Deploying the cluster	1100
8.2.14. Verifying cluster status	1102
8.2.15. Logging in to the cluster	1102
8.2.16. Configuring application access with floating IP addresses	1103
8.2.17. Next steps	1104
8.3. INSTALLING A CLUSTER ON OPENSTACK ON YOUR OWN INFRASTRUCTURE	1104
8.3.1. Prerequisites	1104
8.3.2. Internet and Telemetry access for OpenShift Container Platform	1104
8.3.3. Resource guidelines for installing OpenShift Container Platform on RHOSP	1105
8.3.3.1. Control plane and compute machines	1106
8.3.3.2. Bootstrap machine	1106
8.3.4. Downloading playbook dependencies	1107
8.3.5. Obtaining the installation program	1108
8.3.6. Generating an SSH private key and adding it to the agent	1108
8.3.7. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image	1110
8.3.8. Verifying external network access	1111
8.3.9. Enabling access to the environment	1112
8.3.9.1. Enabling access with floating IP addresses	1112
8.3.10. Defining parameters for the installation program	1113
8.3.11. Creating the installation configuration file	1114
8.3.12. Installation configuration parameters	1116
8.3.12.1. Required configuration parameters	1116
8.3.12.2. Network configuration parameters	1117
8.3.12.3. Optional configuration parameters	1119
8.3.12.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters	1122
8.3.12.5. Optional RHOSP configuration parameters	1123
8.3.12.6. Additional Google Cloud Platform (GCP) configuration parameters	1125
8.3.12.7. Custom subnets in RHOSP deployments	1125
8.3.12.8. Sample customized install-config.yaml file for RHOSP	1126
8.3.12.9. Setting a custom subnet for machines	1127
8.3.12.10. Emptying compute machine pools	1127
8.3.13. Creating the Kubernetes manifest and Ignition config files	1128
8.3.14. Preparing the bootstrap Ignition files	1130
8.3.15. Creating control plane Ignition config files	1133
8.3.16. Creating network resources	1134
8.3.17. Creating the bootstrap machine	1144
8.3.18. Creating the control plane machines	1145
8.3.19. Logging in to the cluster	1148
8.3.20. Deleting bootstrap resources	1148
8.3.21. Creating compute machines	1149
8.3.22. Approving the certificate signing requests for your machines	1151
8.3.23. Verifying a successful installation	1153
8.3.24. Configuring application access with floating IP addresses	1153
8.3.25. Next steps	1154
8.4. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR ON YOUR OWN INFRASTRUCTURE	1154
8.4.1. Prerequisites	1154
8.4.2. About Kuryr SDN	1155
8.4.3. Resource guidelines for installing OpenShift Container Platform on RHOSP with Kuryr	1156
8.4.3.1. Increasing quota	1157
8.4.3.2. Configuring Neutron	1158

8.4.3.3. Configuring Octavia	1158
8.4.3.3.1. The Octavia OVN Driver	1161
8.4.3.4. Known limitations of installing with Kuryr	1162
RHOSP general limitations	1162
RHOSP version limitations	1162
RHOSP environment limitations	1162
RHOSP upgrade limitations	1163
8.4.3.5. Control plane and compute machines	1163
8.4.3.6. Bootstrap machine	1163
8.4.4. Internet and Telemetry access for OpenShift Container Platform	1164
8.4.5. Downloading playbook dependencies	1164
8.4.6. Obtaining the installation program	1165
8.4.7. Generating an SSH private key and adding it to the agent	1166
8.4.8. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image	1167
8.4.9. Verifying external network access	1168
8.4.10. Enabling access to the environment	1169
8.4.10.1. Enabling access with floating IP addresses	1169
8.4.11. Defining parameters for the installation program	1170
8.4.12. Creating the installation configuration file	1171
8.4.13. Installation configuration parameters	1173
8.4.13.1. Required configuration parameters	1173
8.4.13.2. Network configuration parameters	1175
8.4.13.3. Optional configuration parameters	1176
8.4.13.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters	1180
8.4.13.5. Optional RHOSP configuration parameters	1180
8.4.13.6. Additional Google Cloud Platform (GCP) configuration parameters	1182
8.4.13.7. Custom subnets in RHOSP deployments	1182
8.4.13.8. Sample customized install-config.yaml file for RHOSP with Kuryr	1183
8.4.13.9. Setting a custom subnet for machines	1184
8.4.13.10. Emptying compute machine pools	1185
8.4.13.11. Modifying the network type	1185
8.4.14. Creating the Kubernetes manifest and Ignition config files	1186
8.4.15. Preparing the bootstrap Ignition files	1188
8.4.16. Creating control plane Ignition config files	1191
8.4.17. Creating network resources	1191
8.4.18. Creating the bootstrap machine	1201
8.4.19. Creating the control plane machines	1203
8.4.20. Logging in to the cluster	1205
8.4.21. Deleting bootstrap resources	1206
8.4.22. Creating compute machines	1207
8.4.23. Approving the certificate signing requests for your machines	1209
8.4.24. Verifying a successful installation	1211
8.4.25. Configuring application access with floating IP addresses	1211
8.4.26. Next steps	1212
8.5. INSTALLING A CLUSTER ON OPENSTACK IN A RESTRICTED NETWORK	1212
8.5.1. About installations in restricted networks	1213
8.5.1.1. Additional limits	1213
8.5.2. Resource guidelines for installing OpenShift Container Platform on RHOSP	1213
8.5.2.1. Control plane and compute machines	1214
8.5.2.2. Bootstrap machine	1214
8.5.3. Internet and Telemetry access for OpenShift Container Platform	1215
8.5.4. Enabling Swift on RHOSP	1215
8.5.5. Defining parameters for the installation program	1216

8.5.6. Creating the RHCOS image for restricted network installations	1217
8.5.7. Creating the installation configuration file	1219
8.5.7.1. Installation configuration parameters	1221
8.5.7.1.1. Required configuration parameters	1222
8.5.7.1.2. Network configuration parameters	1223
8.5.7.1.3. Optional configuration parameters	1225
8.5.7.1.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters	1229
8.5.7.1.5. Optional RHOSP configuration parameters	1229
8.5.7.1.6. Additional Google Cloud Platform (GCP) configuration parameters	1231
8.5.7.2. Sample customized install-config.yaml file for restricted OpenStack installations	1231
8.5.8. Generating an SSH private key and adding it to the agent	1232
8.5.9. Enabling access to the environment	1234
8.5.9.1. Enabling access with floating IP addresses	1234
8.5.9.2. Enabling access without floating IP addresses	1235
8.5.10. Deploying the cluster	1235
8.5.11. Verifying cluster status	1236
8.5.12. Logging in to the cluster	1237
8.5.13. Configuring application access with floating IP addresses	1238
8.6. UNINSTALLING A CLUSTER ON OPENSTACK	1239
8.6.1. Removing a cluster that uses installer-provisioned infrastructure	1239
8.7. UNINSTALLING A CLUSTER ON OPENSTACK FROM YOUR OWN INFRASTRUCTURE	1239
8.7.1. Prerequisites	1239
8.7.2. Downloading playbook dependencies	1240
8.7.3. Removing a cluster on RHOSP that uses your own infrastructure	1240
CHAPTER 9. INSTALLING ON RHV	1249
9.1. INSTALLING A CLUSTER QUICKLY ON RHV	1249
9.1.1. Prerequisites	1250
9.1.2. Internet and Telemetry access for OpenShift Container Platform	1250
9.1.3. Requirements for the RHV environment	1251
9.1.4. Verifying the requirements for the RHV environment	1252
9.1.5. Preparing the network environment on RHV	1254
9.1.6. Setting up the CA certificate for RHV	1255
9.1.7. Generating an SSH private key and adding it to the agent	1256
9.1.8. Obtaining the installation program	1257
9.1.9. Deploying the cluster	1258
CHAPTER 10. INSTALLING THE CLI BY DOWNLOADING THE BINARY	1262
10.1. INSTALLING THE CLI ON LINUX	1262
10.2. INSTALLING THE CLI ON WINDOWS	1262
10.3. INSTALLING THE CLI ON MACOS	1263
CHAPTER 11. LOGGING IN TO THE CLUSTER	1264
11.1. VERIFYING CLUSTER STATUS	1264
11.2. ACCESSING THE OPENSIFT CONTAINER PLATFORM WEB CONSOLE ON RHV	1265
11.3. TROUBLESHOOTING COMMON ISSUES WITH INSTALLING ON RED HAT VIRTUALIZATION (RHV)	1265
11.3.1. CPU load increases and nodes go into a Not Ready state	1265
11.3.2. Trouble connecting the OpenShift Container Platform cluster API	1266
11.4. POST-INSTALLATION TASKS	1266
11.5. INSTALLING A CLUSTER ON RHV WITH CUSTOMIZATIONS	1266
11.5.1. Prerequisites	1268
11.5.2. Internet and Telemetry access for OpenShift Container Platform	1269
11.5.3. Requirements for the RHV environment	1269
11.5.4. Verifying the requirements for the RHV environment	1270

11.5.5. Preparing the network environment on RHV	1272
11.5.6. Setting up the CA certificate for RHV	1273
11.5.7. Generating an SSH private key and adding it to the agent	1274
11.5.8. Obtaining the installation program	1275
11.5.9. Creating the installation configuration file	1276
11.5.9.1. Example install-config.yaml files for Red Hat Virtualization (RHV)	1279
11.5.9.2. Installation configuration parameters	1281
11.5.9.2.1. Required configuration parameters	1282
11.5.9.2.2. Network configuration parameters	1283
11.5.9.2.3. Optional configuration parameters	1285
11.5.9.2.4. Additional Google Cloud Platform (GCP) configuration parameters	1288
11.5.9.2.5. Additional Red Hat Virtualization (RHV) configuration parameters	1289
11.5.9.2.6. Additional RHV parameters for machine pools	1289
11.5.10. Deploying the cluster	1290
11.5.11. Installing the CLI by downloading the binary	1292
11.5.11.1. Installing the CLI on Linux	1292
11.5.11.2. Installing the CLI on Windows	1293
11.5.11.3. Installing the CLI on macOS	1293
11.5.12. Logging in to the cluster	1293
11.5.13. Verifying cluster status	1294
11.5.14. Accessing the OpenShift Container Platform web console on RHV	1295
11.5.15. Troubleshooting common issues with installing on Red Hat Virtualization (RHV)	1295
11.5.15.1. CPU load increases and nodes go into a Not Ready state	1295
11.5.15.2. Trouble connecting the OpenShift Container Platform cluster API	1296
11.5.16. Post-installation tasks	1296
11.5.17. Next steps	1296
11.6. UNINSTALLING A CLUSTER ON RHV	1296
11.6.1. Removing a cluster that uses installer-provisioned infrastructure	1297
CHAPTER 12. INSTALLING ON VSPHERE	1298
12.1. INSTALLING A CLUSTER ON VSPHERE	1298
12.1.1. Prerequisites	1298
12.1.2. Internet and Telemetry access for OpenShift Container Platform	1298
12.1.3. VMware vSphere infrastructure requirements	1298
12.1.4. vCenter requirements	1299
Required vCenter account privileges	1299
Using OpenShift Container Platform with vMotion	1303
Cluster resources	1304
Cluster limits	1304
Networking requirements	1304
Required IP addresses	1304
DNS records	1304
12.1.5. Generating an SSH private key and adding it to the agent	1305
12.1.6. Obtaining the installation program	1306
12.1.7. Adding vCenter root CA certificates to your system trust	1307
12.1.8. Deploying the cluster	1308
12.1.9. Installing the CLI by downloading the binary	1310
12.1.9.1. Installing the CLI on Linux	1311
12.1.9.2. Installing the CLI on Windows	1311
12.1.9.3. Installing the CLI on macOS	1312
12.1.10. Logging in to the cluster	1312
12.1.11. Creating registry storage	1313
12.1.11.1. Image registry removed during installation	1313

12.1.11.2. Image registry storage configuration	1313
12.1.11.2.1. Configuring registry storage for VMware vSphere	1313
12.1.11.2.2. Configuring block registry storage for VMware vSphere	1315
12.1.12. Backing up VMware vSphere volumes	1316
12.1.13. Next steps	1316
12.2. INSTALLING A CLUSTER ON VSPHERE WITH CUSTOMIZATIONS	1316
12.2.1. Prerequisites	1316
12.2.2. Internet and Telemetry access for OpenShift Container Platform	1317
12.2.3. VMware vSphere infrastructure requirements	1317
12.2.4. vCenter requirements	1318
Required vCenter account privileges	1318
Using OpenShift Container Platform with vMotion	1322
Cluster resources	1323
Cluster limits	1323
Networking requirements	1323
Required IP addresses	1323
DNS records	1323
12.2.5. Generating an SSH private key and adding it to the agent	1324
12.2.6. Obtaining the installation program	1325
12.2.7. Adding vCenter root CA certificates to your system trust	1326
12.2.8. Creating the installation configuration file	1327
12.2.8.1. Installation configuration parameters	1329
12.2.8.1.1. Required configuration parameters	1329
12.2.8.1.2. Network configuration parameters	1331
12.2.8.1.3. Optional configuration parameters	1332
12.2.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters	1336
12.2.8.1.5. Optional VMware vSphere machine pool configuration parameters	1337
12.2.8.2. Sample install-config.yaml file for an installer-provisioned VMware vSphere cluster	1338
12.2.9. Deploying the cluster	1339
12.2.10. Installing the CLI by downloading the binary	1340
12.2.10.1. Installing the CLI on Linux	1341
12.2.10.2. Installing the CLI on Windows	1341
12.2.10.3. Installing the CLI on macOS	1342
12.2.11. Logging in to the cluster	1342
12.2.12. Creating registry storage	1343
12.2.12.1. Image registry removed during installation	1343
12.2.12.2. Image registry storage configuration	1343
12.2.12.2.1. Configuring registry storage for VMware vSphere	1343
12.2.12.2.2. Configuring block registry storage for VMware vSphere	1345
12.2.13. Backing up VMware vSphere volumes	1346
12.2.14. Next steps	1346
12.3. INSTALLING A CLUSTER ON VSPHERE WITH NETWORK CUSTOMIZATIONS	1346
12.3.1. Prerequisites	1347
12.3.2. Internet and Telemetry access for OpenShift Container Platform	1347
12.3.3. VMware vSphere infrastructure requirements	1347
12.3.4. vCenter requirements	1348
Required vCenter account privileges	1348
Using OpenShift Container Platform with vMotion	1352
Cluster resources	1353
Cluster limits	1353
Networking requirements	1353
Required IP addresses	1353
DNS records	1353

12.3.5. Generating an SSH private key and adding it to the agent	1354
12.3.6. Obtaining the installation program	1355
12.3.7. Adding vCenter root CA certificates to your system trust	1356
12.3.8. Creating the installation configuration file	1357
12.3.8.1. Installation configuration parameters	1359
12.3.8.1.1. Required configuration parameters	1359
12.3.8.1.2. Network configuration parameters	1361
12.3.8.1.3. Optional configuration parameters	1362
12.3.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters	1366
12.3.8.2. Network configuration parameters	1366
12.3.8.3. Sample install-config.yaml file for an installer-provisioned VMware vSphere cluster	1367
12.3.9. Modifying advanced network configuration parameters	1368
12.3.10. Cluster Network Operator configuration	1370
12.3.10.1. Configuration parameters for the OpenShift SDN default CNI network provider	1371
12.3.10.2. Cluster Network Operator example configuration	1372
12.3.11. Deploying the cluster	1372
12.3.12. Installing the CLI by downloading the binary	1373
12.3.12.1. Installing the CLI on Linux	1374
12.3.12.2. Installing the CLI on Windows	1374
12.3.12.3. Installing the CLI on macOS	1375
12.3.13. Logging in to the cluster	1375
12.3.14. Creating registry storage	1376
12.3.14.1. Image registry removed during installation	1376
12.3.14.2. Image registry storage configuration	1376
12.3.14.2.1. Configuring registry storage for VMware vSphere	1376
12.3.14.2.2. Configuring block registry storage for VMware vSphere	1378
12.3.15. Backing up VMware vSphere volumes	1379
12.3.16. Next steps	1379
12.4. INSTALLING A CLUSTER ON VSPHERE WITH USER-PROVISIONED INFRASTRUCTURE	1379
12.4.1. Prerequisites	1380
12.4.2. Internet and Telemetry access for OpenShift Container Platform	1380
12.4.3. VMware vSphere infrastructure requirements	1380
12.4.4. Machine requirements for a cluster with user-provisioned infrastructure	1381
12.4.4.1. Required machines	1381
12.4.4.2. Network connectivity requirements	1382
12.4.4.3. Minimum resource requirements	1382
12.4.4.4. Certificate signing requests management	1383
12.4.5. Creating the user-provisioned infrastructure	1383
12.4.5.1. Networking requirements for user-provisioned infrastructure	1383
Network topology requirements	1384
Load balancers	1384
Ethernet adaptor hardware address requirements	1386
12.4.5.2. User-provisioned DNS requirements	1386
12.4.6. Generating an SSH private key and adding it to the agent	1389
12.4.7. Obtaining the installation program	1390
12.4.8. Manually creating the installation configuration file	1391
12.4.8.1. Sample install-config.yaml file for VMware vSphere	1392
12.4.8.2. Configuring the cluster-wide proxy during installation	1394
12.4.9. Creating the Kubernetes manifest and Ignition config files	1395
12.4.10. Extracting the infrastructure name	1398
12.4.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	1398
12.4.12. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	1402
12.4.13. Installing the CLI by downloading the binary	1403

12.4.13.1. Installing the CLI on Linux	1403
12.4.13.2. Installing the CLI on Windows	1403
12.4.13.3. Installing the CLI on macOS	1404
12.4.14. Creating the cluster	1404
12.4.15. Logging in to the cluster	1405
12.4.16. Approving the certificate signing requests for your machines	1406
12.4.17. Initial Operator configuration	1408
12.4.17.1. Image registry removed during installation	1409
12.4.17.2. Image registry storage configuration	1409
12.4.17.2.1. Configuring registry storage for VMware vSphere	1409
12.4.17.2.2. Configuring storage for the image registry in non-production clusters	1411
12.4.17.2.3. Configuring block registry storage for VMware vSphere	1411
12.4.18. Completing installation on user-provisioned infrastructure	1413
12.4.19. Backing up VMware vSphere volumes	1415
12.4.20. Next steps	1415
12.5. INSTALLING A CLUSTER ON VSPHERE WITH NETWORK CUSTOMIZATIONS	1415
12.5.1. Prerequisites	1416
12.5.2. Internet and Telemetry access for OpenShift Container Platform	1416
12.5.3. VMware vSphere infrastructure requirements	1416
12.5.4. Machine requirements for a cluster with user-provisioned infrastructure	1417
12.5.4.1. Required machines	1417
12.5.4.2. Network connectivity requirements	1418
12.5.4.3. Minimum resource requirements	1418
12.5.4.4. Certificate signing requests management	1418
12.5.5. Creating the user-provisioned infrastructure	1419
12.5.5.1. Networking requirements for user-provisioned infrastructure	1419
Network topology requirements	1420
Load balancers	1420
Ethernet adaptor hardware address requirements	1422
12.5.5.2. User-provisioned DNS requirements	1422
12.5.6. Generating an SSH private key and adding it to the agent	1425
12.5.7. Obtaining the installation program	1426
12.5.8. Manually creating the installation configuration file	1427
12.5.8.1. Sample install-config.yaml file for VMware vSphere	1428
12.5.8.2. Network configuration parameters	1430
12.5.9. Modifying advanced network configuration parameters	1430
12.5.10. Cluster Network Operator configuration	1432
12.5.10.1. Configuration parameters for the OpenShift SDN default CNI network provider	1433
12.5.10.2. Cluster Network Operator example configuration	1434
12.5.11. Creating the Ignition config files	1434
12.5.12. Extracting the infrastructure name	1435
12.5.13. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	1436
12.5.14. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	1439
12.5.15. Installing the CLI by downloading the binary	1440
12.5.15.1. Installing the CLI on Linux	1440
12.5.15.2. Installing the CLI on Windows	1441
12.5.15.3. Installing the CLI on macOS	1441
12.5.16. Creating the cluster	1442
12.5.17. Logging in to the cluster	1443
12.5.18. Approving the certificate signing requests for your machines	1443
12.5.19. Initial Operator configuration	1445
12.5.19.1. Image registry removed during installation	1446
12.5.19.2. Image registry storage configuration	1447

12.5.19.2.1. Configuring block registry storage for VMware vSphere	1447
12.5.20. Completing installation on user-provisioned infrastructure	1448
12.5.21. Backing up VMware vSphere volumes	1450
12.5.22. Next steps	1451
12.6. INSTALLING A CLUSTER ON VSPHERE IN A RESTRICTED NETWORK	1451
12.6.1. Prerequisites	1451
12.6.2. About installations in restricted networks	1451
12.6.2.1. Additional limits	1452
12.6.3. Internet and Telemetry access for OpenShift Container Platform	1452
12.6.4. VMware vSphere infrastructure requirements	1452
12.6.5. vCenter requirements	1453
Required vCenter account privileges	1453
Using OpenShift Container Platform with vMotion	1457
Cluster resources	1458
Cluster limits	1458
Networking requirements	1458
Required IP addresses	1458
DNS records	1458
12.6.6. Generating an SSH private key and adding it to the agent	1459
12.6.7. Adding vCenter root CA certificates to your system trust	1460
12.6.8. Creating the RHCOS image for restricted network installations	1461
12.6.9. Creating the installation configuration file	1462
12.6.9.1. Installation configuration parameters	1465
12.6.9.1.1. Required configuration parameters	1465
12.6.9.1.2. Network configuration parameters	1467
12.6.9.1.3. Optional configuration parameters	1468
12.6.9.1.4. Additional Google Cloud Platform (GCP) configuration parameters	1472
12.6.9.1.5. Optional VMware vSphere machine pool configuration parameters	1473
12.6.9.2. Sample install-config.yaml file for an installer-provisioned VMware vSphere cluster	1474
12.6.10. Deploying the cluster	1476
12.6.11. Installing the CLI by downloading the binary	1477
12.6.11.1. Installing the CLI on Linux	1477
12.6.11.2. Installing the CLI on Windows	1478
12.6.11.3. Installing the CLI on macOS	1478
12.6.12. Logging in to the cluster	1479
12.6.13. Creating registry storage	1479
12.6.13.1. Image registry removed during installation	1479
12.6.13.2. Image registry storage configuration	1480
12.6.13.2.1. Configuring registry storage for VMware vSphere	1480
12.6.14. Next steps	1481
12.7. INSTALLING A CLUSTER ON VSPHERE IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE	1481
12.7.1. Prerequisites	1482
12.7.2. About installations in restricted networks	1482
12.7.2.1. Additional limits	1483
12.7.3. Internet and Telemetry access for OpenShift Container Platform	1483
12.7.4. VMware vSphere infrastructure requirements	1483
12.7.5. Machine requirements for a cluster with user-provisioned infrastructure	1484
12.7.5.1. Required machines	1484
12.7.5.2. Network connectivity requirements	1485
12.7.5.3. Minimum resource requirements	1485
12.7.5.4. Certificate signing requests management	1486
12.7.6. Creating the user-provisioned infrastructure	1486

12.7.6.1. Networking requirements for user-provisioned infrastructure	1486
Network topology requirements	1487
Load balancers	1487
Ethernet adaptor hardware address requirements	1489
12.7.6.2. User-provisioned DNS requirements	1489
12.7.7. Generating an SSH private key and adding it to the agent	1492
12.7.8. Manually creating the installation configuration file	1493
12.7.8.1. Sample install-config.yaml file for VMware vSphere	1494
12.7.8.2. Configuring the cluster-wide proxy during installation	1496
12.7.9. Creating the Kubernetes manifest and Ignition config files	1498
12.7.10. Configuring chrony time service	1500
12.7.11. Extracting the infrastructure name	1501
12.7.12. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	1502
12.7.13. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere	1505
12.7.14. Creating the cluster	1506
12.7.15. Logging in to the cluster	1507
12.7.16. Approving the certificate signing requests for your machines	1508
12.7.17. Initial Operator configuration	1510
12.7.17.1. Image registry storage configuration	1511
12.7.17.1.1. Configuring registry storage for VMware vSphere	1511
12.7.17.1.2. Configuring storage for the image registry in non-production clusters	1512
12.7.17.1.3. Configuring block registry storage for VMware vSphere	1513
12.7.18. Completing installation on user-provisioned infrastructure	1514
12.7.19. Backing up VMware vSphere volumes	1517
12.7.20. Next steps	1517
12.8. UNINSTALLING A CLUSTER ON VSPHERE THAT USES INSTALLER-PROVISIONED INFRASTRUCTURE	1517
12.8.1. Removing a cluster that uses installer-provisioned infrastructure	1517
CHAPTER 13. INSTALLATION CONFIGURATION	1519
13.1. SUPPORTED INSTALLATION METHODS FOR DIFFERENT PLATFORMS	1519
13.2. CUSTOMIZING NODES	1520
13.2.1. Adding day-1 kernel arguments	1520
13.2.2. Adding kernel modules to nodes	1521
13.2.2.1. Building and testing the kernel module container	1522
13.2.2.2. Provisioning a kernel module to OpenShift Container Platform	1524
13.2.2.2.1. Provision kernel modules via a MachineConfig object	1525
13.2.3. Encrypting disks during installation	1528
13.2.3.1. Enabling TPM v2 disk encryption	1528
13.2.3.2. Enabling Tang disk encryption	1529
13.2.4. Configuring chrony time service	1532
13.2.5. Additional resources	1533
13.3. AVAILABLE CLUSTER CUSTOMIZATIONS	1533
13.3.1. Cluster configuration resources	1534
13.3.2. Operator configuration resources	1534
13.3.3. Additional configuration resources	1535
13.3.4. Informational Resources	1535
13.3.5. Updating the global cluster pull secret	1536
13.4. CONFIGURING YOUR FIREWALL	1537
13.4.1. Configuring your firewall for OpenShift Container Platform	1537
13.5. CONFIGURING A PRIVATE CLUSTER	1540
13.5.1. About private clusters	1540
DNS	1540

Ingress Controller	1540
API server	1540
13.5.2. Setting DNS to private	1540
13.5.3. Setting the Ingress Controller to private	1542
13.5.4. Restricting the API server to private	1542
CHAPTER 14. TROUBLESHOOTING INSTALLATION ISSUES	1544
14.1. PREREQUISITES	1544
14.2. GATHERING LOGS FROM A FAILED INSTALLATION	1544
14.3. MANUALLY GATHERING LOGS WITH SSH ACCESS TO YOUR HOST(S)	1545
14.4. MANUALLY GATHERING LOGS WITHOUT SSH ACCESS TO YOUR HOST(S)	1546
14.5. GETTING DEBUG INFORMATION FROM THE INSTALLATION PROGRAM	1546
CHAPTER 15. SUPPORT FOR FIPS CRYPTOGRAPHY	1548
15.1. FIPS VALIDATION IN OPENSIFT CONTAINER PLATFORM	1548
15.2. FIPS SUPPORT IN COMPONENTS THAT THE CLUSTER USES	1548
15.2.1. etcd	1548
15.2.2. Storage	1549
15.2.3. Runtimes	1549
15.3. INSTALLING A CLUSTER IN FIPS MODE	1549

CHAPTER 1. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION

Before you install a cluster on infrastructure that you provision in a restricted network, you must mirror the required container images into that environment. You can also use this procedure in unrestricted networks to ensure your clusters only use container images that have satisfied your organizational controls on external content.



IMPORTANT

You must have access to the internet to obtain the necessary container images. In this procedure, you place the mirror registry on a mirror host that has access to both your network and the Internet. If you do not have access to a mirror host, use the disconnected procedure to copy images to a device you can move across network boundaries with.

1.1. PREREQUISITES

- You must have a container image registry that supports [Docker v2-2](#) in the location that will host the OpenShift Container Platform cluster, such as one of the following registries:
 - [Red Hat Quay](#)
 - [JFrog Artifactory](#)
 - [Sonatype Nexus Repository](#)
 - [Harbor](#)

If you have an entitlement to Red Hat Quay, see the documentation on deploying Red Hat Quay [for proof-of-concept purposes](#) or [by using the Quay Operator](#). If you need additional assistance selecting and installing a registry, contact your sales representative or Red Hat support.

1.2. ABOUT THE MIRROR REGISTRY

You can mirror the images that are required for OpenShift Container Platform installation and subsequent product updates to a mirror registry. These actions use the same process. The release image, which contains the description of the content, and the images it references are all mirrored. In addition, the Operator catalog source image and the images that it references must be mirrored for each Operator that you use. After you mirror the content, you configure each cluster to retrieve this content from your mirror registry.

The mirror registry can be any container registry that supports [Docker v2-2](#). All major cloud provider registries, as well as Red Hat Quay, Artifactory, and others, have the necessary support. Using one of these registries ensures that OpenShift Container Platform can verify the integrity of each image in disconnected environments.

The mirror registry must be reachable by every machine in the clusters that you provision. If the registry is unreachable installation, updating, or normal operations such as workload relocation might fail. For that reason, you must run mirror registries in a highly available way, and the mirror registries must at least match the production availability of your OpenShift Container Platform clusters.

When you populate a mirror registry with OpenShift Container Platform images, you can follow two scenarios. If you have a host that can access both the internet and your mirror registry, but not your cluster nodes, you can directly mirror the content from that machine. This process is referred to as

connected mirroring. If you have no such host, you must mirror the images to a file system and then bring that host or removable media into your restricted environment. This process is referred to as *disconnected mirroring*.

1.3. PREPARING YOUR MIRROR HOST

Before you perform the mirror procedure, you must prepare the host to retrieve content and push it to the remote location.

1.3.1. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

1.3.1.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.3.1.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.

2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

1.3.1.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

1.4. CONFIGURING CREDENTIALS THAT ALLOW IMAGES TO BE MIRRORED

Create a container image registry credentials file that allows mirroring images from Red Hat to your mirror.

**WARNING**

Do not use this image registry credentials file as the pull secret when you install a cluster. If you provide this file when you install cluster, all of the machines in the cluster will have write access to your mirror registry.

**WARNING**

This process requires that you have write access to a container image registry on the mirror registry and adds the credentials to a registry pull secret.

Prerequisites

- You configured a mirror registry to use in your restricted network.
- You identified an image repository location on your mirror registry to mirror images into.
- You provisioned a mirror registry account that allows images to be uploaded to that image repository.

Procedure

Complete the following steps on the installation host:

1. Download your **registry.redhat.io** pull secret from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site and save it to a **.json** file.
2. Generate the base64-encoded user name and password or token for your mirror registry:

```
$ echo -n '<user_name>:<password>' | base64 -w0 1
BGVtbYk3ZHAtdXs=
```

- 1 For **<user_name>** and **<password>**, specify the user name and password that you configured for your registry.

3. Make a copy of your pull secret in JSON format:

```
$ cat ./pull-secret.text | jq . > <path>/<pull-secret-file> 1
```

- 1 Specify the path to the folder to store the pull secret in and a name for the JSON file that you create.

The contents of the file resemble the following example:

```
{
```

```

"auths": {
  "cloud.openshift.com": {
    "auth": "b3BlbnNo...",
    "email": "you@example.com"
  },
  "quay.io": {
    "auth": "b3BlbnNo...",
    "email": "you@example.com"
  },
  "registry.connect.redhat.com": {
    "auth": "NTE3Njg5Nj...",
    "email": "you@example.com"
  },
  "registry.redhat.io": {
    "auth": "NTE3Njg5Nj...",
    "email": "you@example.com"
  }
}
}
}

```

4. Edit the new file and add a section that describes your registry to it:

```

"auths": {
  "<mirror_registry>": { 1
    "auth": "<credentials>", 2
    "email": "you@example.com"
  },

```

- 1** For **<mirror_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:5000**
- 2** For **<credentials>**, specify the base64-encoded user name and password for the mirror registry.

The file resembles the following example:

```

{
  "auths": {
    "<mirror_registry>": {
      "auth": "<credentials>",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}

```

```

    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}

```

1.5. MIRRORING THE OPENSIFT CONTAINER PLATFORM IMAGE REPOSITORY

Mirror the OpenShift Container Platform image repository to your registry to use during cluster installation or upgrade.

Prerequisites

- Your mirror host has access to the Internet.
- You configured a mirror registry to use in your restricted network and can access the certificate and credentials that you configured.
- You downloaded the pull secret from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site and modified it to include authentication to your mirror repository.
- If you use self-signed certificates that do not set a Subject Alternative Name, you must precede the **oc** commands in this procedure with **GODEBUG=x509ignoreCN=0**. If you do not set this variable, the **oc** commands will fail with the following error:

```
x509: certificate relies on legacy Common Name field, use SANs or temporarily enable
Common Name matching with GODEBUG=x509ignoreCN=0
```

Procedure

Complete the following steps on the mirror host:

1. Review the [OpenShift Container Platform downloads page](#) to determine the version of OpenShift Container Platform that you want to install and determine the corresponding tag on the [Repository Tags](#) page.
2. Set the required environment variables:
 - a. Export the release version:

```
$ OCP_RELEASE=<release_version>
```

For **<release_version>**, specify the tag that corresponds to the version of OpenShift Container Platform to install, such as **4.5.4**.

- b. Export the local registry name and host port:

```
$ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
```

For **<local_registry_host_name>**, specify the registry domain name for your mirror repository, and for **<local_registry_host_port>**, specify the port that it serves content on.

- c. Export the local repository name:

```
$ LOCAL_REPOSITORY='<local_repository_name>'
```

For **<local_repository_name>**, specify the name of the repository to create in your registry, such as **ocp4/openshift4**.

- d. Export the name of the repository to mirror:

```
$ PRODUCT_REPO='openshift-release-dev'
```

For a production release, you must specify **openshift-release-dev**.

- e. Export the path to your registry pull secret:

```
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

For **<path_to_pull_secret>**, specify the absolute path to and file name of the pull secret for your mirror registry that you created.

- f. Export the release mirror:

```
$ RELEASE_NAME="ocp-release"
```

For a production release, you must specify **ocp-release**.

- g. Export the type of architecture for your server, such as **x86_64**:

```
$ ARCHITECTURE=<server_architecture>
```

- h. Export the path to the directory to host the mirrored images:

```
$ REMOVABLE_MEDIA_PATH=<path> 1
```

- 1** Specify the full path, including the initial forward slash (/) character.

3. Mirror the version images to the internal container registry:

- If your mirror host does not have Internet access, take the following actions:
 - i. Connect the removable media to a system that is connected to the Internet.
 - ii. Review the images and configuration manifests to mirror:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} \
  --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
  ${ARCHITECTURE} \
  --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
  --to-release-
  image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
  ${ARCHITECTURE} --dry-run
```

- iii. Record the entire **imageContentSources** section from the output of the previous command. The information about your mirrors is unique to your mirrored repository, and

you must add the **imageContentSources** section to the **install-config.yaml** file during installation.

- iv. Mirror the images to a directory on the removable media:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-dir=${REMOVABLE_MEDIA_PATH}/mirror quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE}
```

- v. Take the media to the restricted network environment and upload the images to the local container registry.

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-dir=${REMOVABLE_MEDIA_PATH}/mirror "file://openshift/release:${OCP_RELEASE}*" ${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} 1
```

- 1** For **REMOVABLE_MEDIA_PATH**, you must use the same path that you specified when you mirrored the images.

- If the local container registry is connected to the mirror host, take the following actions:
 - i. Directly push the release images to the local registry by using following command:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} \ --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} \ --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \ --to-release-image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}
```

This command pulls the release information as a digest, and its output includes the **imageContentSources** data that you require when you install your cluster.

- ii. Record the entire **imageContentSources** section from the output of the previous command. The information about your mirrors is unique to your mirrored repository, and you must add the **imageContentSources** section to the **install-config.yaml** file during installation.



NOTE

The image name gets patched to Quay.io during the mirroring process, and the podman images will show Quay.io in the registry on the bootstrap virtual machine.

4. To create the installation program that is based on the content that you mirrored, extract it and pin it to the release:

- If your mirror host does not have Internet access, run the following command:

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install "${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
```

-
- If the local container registry is connected to the mirror host, run the following command:

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install  
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-  
${ARCHITECTURE}"
```



IMPORTANT

To ensure that you use the correct images for the version of OpenShift Container Platform that you selected, you must extract the installation program from the mirrored content.

You must perform this step on a machine with an active Internet connection.

If you are in a disconnected environment, use the **--image** flag as part of `must-gather` and point to the payload image.

1.6. THE CLUSTER SAMPLES OPERATOR IN A DISCONNECTED ENVIRONMENT

In a disconnected environment, you must take additional steps after you install a cluster to configure the Cluster Samples Operator. == Next steps

- [Mirror](#) the OperatorHub images for the Operators that you want to install in your cluster.
- Install a cluster on infrastructure that you provision in your restricted network, such as on [VMware vSphere](#), [bare metal](#), or [Amazon Web Services](#).

1.7. ADDITIONAL RESOURCES

- See [Gathering data about specific features](#) for more information about using `must-gather`.

CHAPTER 2. INSTALLING ON AWS

2.1. CONFIGURING AN AWS ACCOUNT

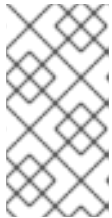
Before you can install OpenShift Container Platform, you must configure an Amazon Web Services (AWS) account.

2.1.1. Configuring Route 53

To install OpenShift Container Platform, the Amazon Web Services (AWS) account you use must have a dedicated public hosted zone in your Route 53 service. This zone must be authoritative for the domain. The Route 53 service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through AWS or another source.



NOTE

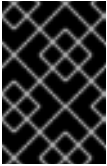
If you purchase a new domain through AWS, it takes time for the relevant DNS changes to propagate. For more information about purchasing domains through AWS, see [Registering Domain Names Using Amazon Route 53](#) in the AWS documentation.

2. If you are using an existing domain and registrar, migrate its DNS to AWS. See [Making Amazon Route 53 the DNS Service for an Existing Domain](#) in the AWS documentation.
3. Create a public hosted zone for your domain or subdomain. See [Creating a Public Hosted Zone](#) in the AWS documentation.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
4. Extract the new authoritative name servers from the hosted zone records. See [Getting the Name Servers for a Public Hosted Zone](#) in the AWS documentation.
5. Update the registrar records for the AWS Route 53 name servers that your domain uses. For example, if you registered your domain to a Route 53 service in a different accounts, see the following topic in the AWS documentation: [Adding or Changing Name Servers or Glue Records](#).
6. If you are using a subdomain, add its delegation records to the parent domain. This gives Amazon Route 53 responsibility for the subdomain. Follow the delegation procedure outlined by the DNS provider of the parent domain. See [Creating a subdomain that uses Amazon Route 53 as the DNS service without migrating the parent domain](#) in the AWS documentation for an example high level procedure.

2.1.2. AWS account limits

The OpenShift Container Platform cluster uses a number of Amazon Web Services (AWS) components, and the default [Service Limits](#) affect your ability to install OpenShift Container Platform clusters. If you use certain cluster configurations, deploy your cluster in certain AWS regions, or run multiple clusters from your account, you might need to request additional resources for your AWS account.

The following table summarizes the AWS components whose limits can impact your ability to install and run OpenShift Container Platform clusters.

Component	Number of clusters available by default	Default AWS limit	Description
Instance Limits	Varies	Varies	<p>By default, each cluster creates the following instances:</p> <ul style="list-style-type: none"> • One bootstrap machine, which is removed after installation • Three master nodes • Three worker nodes <p>These instance type counts are within a new account's default limit. To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, review your account limits to ensure that your cluster can deploy the machines that you need.</p> <p>In most regions, the bootstrap and worker machines use an m4.large machines and the master machines use m4.xlarge instances. In some regions, including all regions that do not support these instance types, m5.large and m5.xlarge instances are used instead.</p>
Elastic IPs (EIPs)	0 to 1	5 EIPs per account	<p>To provision the cluster in a highly available configuration, the installation program creates a public and private subnet for each availability zone within a region. Each private subnet requires a NAT Gateway, and each NAT gateway requires a separate elastic IP. Review the AWS region map to determine how many availability zones are in each region. To take advantage of the default high availability, install the cluster in a region with at least three availability zones. To install a cluster in a region with more than five availability zones, you must increase the EIP limit.</p> <div style="display: flex; align-items: flex-start; margin-top: 20px;">  <div> <p>IMPORTANT</p> <p>To use the us-east-1 region, you must increase the EIP limit for your account.</p> </div> </div>

Component	Number of clusters available by default	Default AWS limit	Description
Virtual Private Clouds (VPCs)	5	5 VPCs per region	Each cluster creates its own VPC.
Elastic Load Balancing (ELB/NLB)	3	20 per region	By default, each cluster creates internal and external network load balancers for the master API server and a single classic elastic load balancer for the router. Deploying more Kubernetes Service objects with type LoadBalancer will create additional load balancers .
NAT Gateways	5	5 per availability zone	The cluster deploys one NAT gateway in each availability zone.
Elastic Network Interfaces (ENIs)	At least 12	350 per region	The default installation creates 21 ENIs and an ENI for each availability zone in your region. For example, the us-east-1 region contains six availability zones, so a cluster that is deployed in that zone uses 27 ENIs. Review the AWS region map to determine how many availability zones are in each region. Additional ENIs are created for additional machines and elastic load balancers that are created by cluster usage and deployed workloads.
VPC Gateway	20	20 per account	Each cluster creates a single VPC Gateway for S3 access.
S3 buckets	99	100 buckets per account	Because the installation process creates a temporary bucket and the registry component in each cluster creates a bucket, you can create only 99 OpenShift Container Platform clusters per AWS account.
Security Groups	250	2,500 per account	Each cluster creates 10 distinct security groups.

2.1.3. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 2.1. Required EC2 permissions for installation

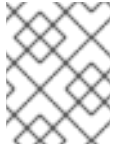
- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2>DeleteSecurityGroup**
- **ec2>DeleteSnapshot**
- **ec2>DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**

- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 2.2. Required permissions for creating network resources during installation

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**

- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



NOTE

If you use an existing VPC, your account does not require these permissions for creating network resources.

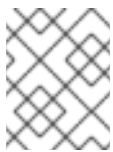
Example 2.3. Required Elastic Load Balancing permissions for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing>CreateListener**
- **elasticloadbalancing>CreateLoadBalancer**
- **elasticloadbalancing>CreateLoadBalancerListeners**
- **elasticloadbalancing>CreateTargetGroup**
- **elasticloadbalancing>DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**

- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

Example 2.4. Required IAM permissions for installation

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam>DeleteRole**
- **iam>DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam>ListInstanceProfilesForRole**
- **iam>ListRoles**
- **iam>ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



NOTE

If you have not created an elastic load balancer (ELB) in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 2.5. Required Route 53 permissions for installation

- **route53:ChangeResourceRecordSets**

- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53>DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Example 2.6. Required S3 permissions for installation

- **s3:CreateBucket**
- **s3>DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**

- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

Example 2.7. S3 permissions that cluster Operators require

- **s3>DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

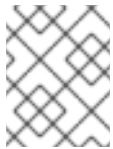
Example 2.8. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2>DeleteNetworkInterface**
- **ec2>DeleteVolume**
- **elasticloadbalancing>DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3>DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 2.9. Required permissions to delete network resources

- **ec2>DeleteDhcpOptions**

- **ec2:DeleteInternetGateway**
- **ec2>DeleteNatGateway**
- **ec2>DeleteRoute**
- **ec2>DeleteRouteTable**
- **ec2>DeleteSubnet**
- **ec2>DeleteVpc**
- **ec2>DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**

**NOTE**

If you use an existing VPC, your account does not require these permissions to delete network resources.

Example 2.10. Additional IAM and S3 permissions that are required to create manifests

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam>DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam>ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam>ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**

- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

2.1.4. Creating an IAM user

Each Amazon Web Services (AWS) account contains a root user account that is based on the email address you used to create the account. This is a highly-privileged account, and it is recommended to use it for only initial account and billing configuration, creating an initial set of users, and securing the account.

Before you install OpenShift Container Platform, create a secondary IAM administrative user. As you complete the [Creating an IAM User in Your AWS Account](#) procedure in the AWS documentation, set the following options:

Procedure

1. Specify the IAM user name and select **Programmatic access**.
2. Attach the **AdministratorAccess** policy to ensure that the account has sufficient permission to create the cluster. This policy provides the cluster with the ability to grant credentials to each OpenShift Container Platform component. The cluster grants the components only the credentials that they require.



NOTE

While it is possible to create a policy that grants the all of the required AWS permissions and attach it to the user, this is not the preferred option. The cluster will not have the ability to grant additional credentials to individual components, so the same credentials are used by all components.

3. Optional: Add metadata to the user by attaching tags.
4. Confirm that the user name that you specified is granted the **AdministratorAccess** policy.
5. Record the access key ID and secret access key values. You must use these values when you configure your local machine to run the installation program.



IMPORTANT

You cannot use a temporary session token that you generated while using a multi-factor authentication device to authenticate to AWS when you deploy a cluster. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials.

2.1.5. Supported AWS regions

You can deploy an OpenShift Container Platform cluster to the following regions:

- **ap-northeast-1** (Tokyo)
- **ap-northeast-2** (Seoul)

- **ap-south-1** (Mumbai)
- **ap-southeast-1** (Singapore)
- **ap-southeast-2** (Sydney)
- **ca-central-1** (Central)
- **eu-central-1** (Frankfurt)
- **eu-north-1** (Stockholm)
- **eu-west-1** (Ireland)
- **eu-west-2** (London)
- **eu-west-3** (Paris)
- **me-south-1** (Bahrain)
- **sa-east-1** (São Paulo)
- **us-east-1** (N. Virginia)
- **us-east-2** (Ohio)
- **us-west-1** (N. California)
- **us-west-2** (Oregon)

2.1.6. Next steps

- Install an OpenShift Container Platform cluster:
 - [Quickly install a cluster](#) with default options on installer-provisioned infrastructure
 - [Install a cluster with cloud customizations on installer-provisioned infrastructure](#)
 - [Install a cluster with network customizations on installer-provisioned infrastructure](#)
 - [Installing a cluster on user-provisioned infrastructure in AWS by using CloudFormation templates](#)

2.2. MANUALLY CREATING IAM FOR AWS

2.2.1. Manually create IAM

The Cloud Credential Operator can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Run the OpenShift Container Platform installer to generate manifests:

```
$ openshift-install create manifests --dir=mycluster
```

2. Insert a config map into the manifests directory so that the Cloud Credential Operator is placed in manual mode:

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-credential-operator-config
  namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF
```

3. Remove the **admin** credential secret created using your local cloud credentials. This removal prevents your **admin** credential from being stored in the cluster:

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. Obtain the OpenShift Container Platform release image your **openshift-install** binary is built to use:

```
$ bin/openshift-install version
```

Example output

```
release image quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64
```

5. Locate all **CredentialsRequest** objects in this release image that target the cloud you are deploying on:

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64 --to
./release-image
```

6. Locate the **CredentialsRequests** in the extracted file:

```
$ grep -l "apiVersion: cloudcredential.openshift.io" * | xargs cat
```



NOTE

In a future OpenShift Container Platform release, there will be a new **oc adm release** command to scan for the **CredentialsRequests** and display them.

This displays the details for each request. Remember to ignore any **CredentialsRequests** where the **spec.providerSpec.kind** does not match the cloud provider you are installing to.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
```

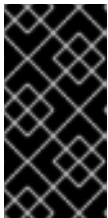
```

metadata:
  name: cloud-credential-operator-iam-ro
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: cloud-credential-operator-iam-ro-creds
    namespace: openshift-cloud-credential-operator
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - effect: Allow
        action:
          - iam:GetUser
          - iam:GetUserPolicy
          - iam:ListAccessKeys
        resource: "*"

```

7. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in each **request.spec.secretRef**. The format for the secret data varies for each cloud provider.
8. Proceed with cluster creation:

```
$ openshift-install create cluster --dir=mycluster
```



IMPORTANT

Before performing an upgrade, you might need to adjust your credentials if permissions have changed in the next release. In the future, the Cloud Credential Operator might prevent you from upgrading until you have indicated that you have addressed updated permissions.

2.2.2. Admin credentials root secret format

Each cloud provider uses a credentials root secret in the **kube-system** namespace by convention, which is then used to satisfy all credentials requests and create their respective secrets. This is done either by minting new credentials, *Mint Mode*, or by copying the credentials root secret, *Passthrough Mode*.

The format for the secret varies by cloud, and is also used for each **CredentialsRequest** secret.

Amazon Web Services (AWS) secret format

```

apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: aws-creds
stringData:
  aws_access_key_id: <AccessKeyID>
  aws_secret_access_key: <SecretAccessKey>

```

2.2.2.1. Upgrades

In a future release, improvements to the Cloud Credential Operator will prevent situations where a user might enter an upgrade that will fail because their manually maintained credentials have not been updated to match the **CredentialsRequest** objects in the upcoming release image.

2.2.3. Mint Mode

Mint Mode is supported for AWS, GCP, and Azure.

The default and recommended best practice for running OpenShift Container Platform is to run the installer with an administrator-level cloud credential. The **admin** credential is stored in the **kube-system** namespace, and then used by the Cloud Credential Operator to process the **CredentialsRequest** objects in the cluster and create new users for each with specific permissions.

The benefits of Mint Mode include:

- Each cluster component only has the permissions it requires.
- Automatic, on-going reconciliation for cloud credentials including upgrades, which might require additional credentials or permissions.

One drawback is that Mint Mode requires **admin** credential storage in a cluster **kube-system** secret.

2.2.4. Mint Mode with removal or rotation of the admin credential

Currently, this mode is only supported on AWS.

In this mode, a user installs OpenShift Container Platform with an **admin** credential just like the normal mint mode. However, this mode removes the **admin** credential secret from the cluster post-installation.

The administrator can have the Cloud Credential Operator make its own request for a read-only credential that allows it to verify if all **CredentialsRequest** objects have their required permissions, thus the **admin** credential is not required unless something needs to be changed. After the associated credential is removed, it can be destroyed on the underlying cloud, if desired.

Prior to upgrade, the **admin** credential should be restored. In the future, upgrade might be blocked if the credential is not present.

The **admin** credential is not stored in the cluster permanently.

This mode still requires the **admin** credential in the cluster for brief periods of time. It also requires manually re-instating the secret with **admin** credentials for each upgrade.

2.3. INSTALLING A CLUSTER QUICKLY ON AWS

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) that uses the default configuration options.

2.3.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.3.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

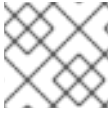


IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.3.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

Identity added: /home/<you>/<path>/<file_name> (<computer_name>)

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

2.3.4. Obtaining the installation program

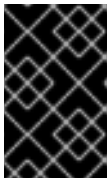
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

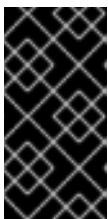
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a `.txt` file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.3.5. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

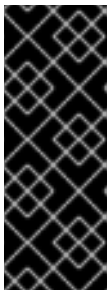
- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

- a. Optional: Select an SSH key to use to access your cluster machines.

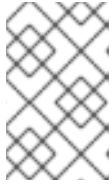


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- b. Select **aws** as the platform to target.
- c. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
- d. Select the AWS region to deploy the cluster to.

- e. Select the base domain for the Route 53 service that you configured for your cluster.
- f. Enter a descriptive name for your cluster.
- g. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.

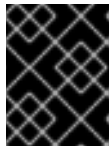
**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

2.3.6. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.3.6.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.

2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.3.6.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.3.6.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.

3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.3.7. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

2.3.8. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- If necessary, you can [remove cloud provider credentials](#).

2.4. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a customized cluster on infrastructure that the installation program provisions on Amazon Web Services (AWS). To customize the installation, you modify parameters in the `install-config.yaml` file before you install the cluster.

2.4.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

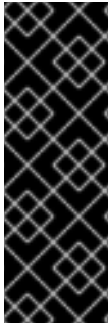
2.4.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

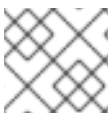


IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.4.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:


```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

2.4.4. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

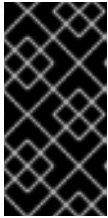
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.4.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

2.4.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

2.4.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 2.1. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object


Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

2.4.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 2.2. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


2.4.5.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 2.3. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;"> <div style="background-color: black; width: 20px; height: 100px; margin-right: 10px;"></div> <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

2.4.5.1.4. Optional AWS configuration parameters

Optional AWS configuration parameters are described in the following table:

Table 2.4. Optional AWS parameters

Parameter	Description	Values
compute.platform.aws.rootVolume.iops	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
compute.platform.aws.rootVolume.size	The size in GiB of the root volume.	Integer, for example 500 .
compute.platform.aws.rootVolume.type	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
compute.platform.aws.type	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .
compute.platform.aws.zones	The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
compute.aws.region	The AWS region that the installation program creates compute resources in.	Any valid AWS region , such as us-east-1 .

Parameter	Description	Values
controlPlane.platform.aws.type	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
controlPlane.platform.aws.zones	The availability zones where the installation program creates machines for the control plane machine pool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
controlPlane.aws.region	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
platform.aws.userTags	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.
platform.aws.subnets	If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone.	Valid subnet IDs.

2.4.5.2. Sample customized `install-config.yaml` file for AWS

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

apiVersion: v1

baseDomain: example.com **1**

```

controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    aws:
      zones:
        - us-west-2a
        - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
        type: m5.xlarge 5
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
        type: c5.4xlarge
      zones:
        - us-west-2c
      replicas: 3
metadata:
  name: test-cluster 9
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 10
    userTags:
      adminContact: jdoe
      costCenter: 7536
  fips: false 11
  sshKey: ssh-ed25519 AAAA... 12
  pullSecret: '{"auths": ...}' 13

```

1 9 10 13 Required. The installation program prompts you for this value.

2 6 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section

must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

- 4 5** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

- 8** To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 11** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 12** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

2.4.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

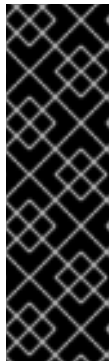
- 1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

2.4.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.4.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.4.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.4.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.

2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.4.8. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

2.4.9. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

- If necessary, you can [remove cloud provider credentials](#).

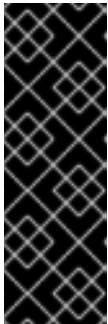
2.5. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

2.5.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.5.2. Internet and Telemetry access for OpenShift Container Platform

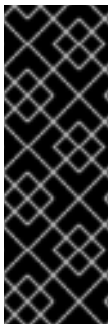
In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.

- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.5.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

2.5.4. Obtaining the installation program

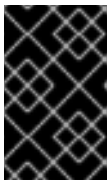
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.5.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

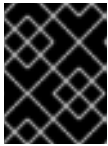
Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

2.5.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

2.5.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 2.5. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object

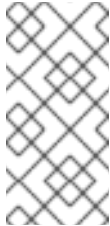
Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

2.5.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 2.6. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


2.5.5.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 2.7. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;"> <div style="background-color: black; width: 20px; height: 100px; margin-right: 10px;"></div> <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

2.5.5.1.4. Optional AWS configuration parameters

Optional AWS configuration parameters are described in the following table:

Table 2.8. Optional AWS parameters

Parameter	Description	Values
compute.platform.aws.rootVolume.iops	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
compute.platform.aws.rootVolume.size	The size in GiB of the root volume.	Integer, for example 500 .
compute.platform.aws.rootVolume.type	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
compute.platform.aws.type	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .
compute.platform.aws.zones	The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
compute.aws.region	The AWS region that the installation program creates compute resources in.	Any valid AWS region , such as us-east-1 .

Parameter	Description	Values
controlPlane.platform.aws.type	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
controlPlane.platform.aws.zones	The availability zones where the installation program creates machines for the control plane machine pool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
controlPlane.aws.region	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
platform.aws.userTags	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.
platform.aws.subnets	If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone.	Valid subnet IDs.



IMPORTANT

The Open Virtual Networking (OVN) Kubernetes network plug-in is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of the OVN Technology Preview, see <https://access.redhat.com/articles/4380121>.

2.5.5.2. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 2.9. Required network parameters

Parameter	Description	Value
networking.net workType	The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5. The OVNKubernetes plug-in is available as a Technology Preview in OpenShift Container Platform 4.5.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clus terNetwork[].cid r	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .
networking.clus terNetwork[].ho stPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serv iceNetwork[]	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.mac hineNetwork[].ci dr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

2.5.5.3. Sample customized install-config.yaml file for AWS

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
    rootVolume:
      iops: 4000
      size: 500
      type: io1
      type: m5.xlarge 5
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
        type: c5.4xlarge
      zones:
      - us-west-2c
    replicas: 3
  metadata:
    name: test-cluster 9
networking: 10
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
  fips: false 12
  sshKey: ssh-ed25519 AAAA... 13
  pullSecret: '{"auths": ...}' 14

```

1 9 11 14 Required. The installation program prompts you for this value.

2 6 10 If you do not provide these parameters and values, the installation program provides the default value.

- 3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section
- 4 5 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

- 8 To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iopts** to **2000**.
- 12 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 13 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

2.5.6. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster. Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of [kube-proxy](#) settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.

Procedure

1. Use the following command to create manifests:


```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1 For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, several network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

- 1 The parameters for the **spec** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.

- Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

2.5.7. Cluster Network Operator configuration

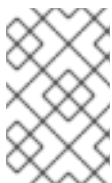
The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
  ...
  kubeProxyConfig: 4
  iptablesSyncPeriod: 30s 5
  proxyArguments:
    iptables-min-sync-period: 6
    - 0s
```

- 1 2 Specified in the **install-config.yaml** file.
- 3 Configures the default Container Network Interface (CNI) network provider for the cluster network.
- 4 The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.
- 5 The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.



NOTE

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

- 6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh

does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#).

2.5.7.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
  mode: NetworkPolicy 3
  mtu: 1450 4
  vxlanPort: 4789 5
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**.

- 5 The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for VXLAN, since both SDNs use the same default VXLAN port number.

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

2.5.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider

The following YAML object describes the configuration parameters for the OVN-Kubernetes default CNI network provider.

```
defaultNetwork:
  type: OVNKubernetes 1
  ovnKubernetesConfig: 2
  mtu: 1400 3
  genevePort: 6081 4
```

■

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OVN-Kubernetes configuration.
- 3 The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **100** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1400**.

- 4 The UDP port for the Geneve overlay network.

2.5.7.3. Cluster Network Operator example configuration

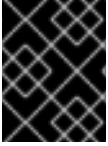
A complete CR object for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 0s
```

2.5.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

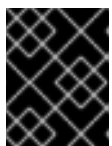
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



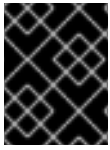
IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

2.5.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.5.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.5.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

-

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.5.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.5.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

2.5.11. Next steps

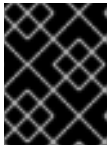
- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- If necessary, you can [remove cloud provider credentials](#).

2.6. INSTALLING A CLUSTER ON AWS IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) in a restricted network by creating an internal mirror of the installation release content on an existing Amazon Virtual Private Cloud (VPC).

2.6.1. Prerequisites

- You [mirrored the images for a disconnected installation](#) to your registry and obtained the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- You have an existing VPC in AWS. When installing to a restricted network using installer-provisioned infrastructure, you cannot use the installer-provisioned VPC. You must use a user-provisioned VPC that satisfies one of the following requirements:
 - Contains the mirror registry.
 - Has firewall rules or a peering connection to access the mirror registry hosted elsewhere.
- You reviewed details about the [OpenShift Container Platform installation and update](#) processes.
- You [configured an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- You downloaded the AWS CLI and installed it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#) in the AWS documentation.
- If you use a firewall and plan to use the Telemetry service, you [configured the firewall to allow the sites](#) that your cluster requires access to.



NOTE

If you are configuring a proxy, be sure to also review this site list.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.6.2. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.

2.6.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

2.6.3. About using a custom VPC

In OpenShift Container Platform 4.5, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift

Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

2.6.3.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VPC options like DHCP, so you must do so before you install the cluster.

Your VPC must meet the following characteristics:

- The VPC's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VPC must not use the **kubernetes.io/cluster/.*: owned** tag.
- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See [DNS Support in Your VPC](#) in the AWS documentation.

If you use a cluster with public access, you must create a public and a private subnet for each availability zone that your cluster uses.

The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. Review the current [Tag Restrictions](#) in the AWS documentation to ensure that the installation program can add a tag to each subnet that you specify.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**

- `elasticloadbalancing.<region>.amazonaws.com`
- `s3.<region>.amazonaws.com`

Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

Component	AWS type	Description								
VPC	<ul style="list-style-type: none"> • AWS::EC2::VPC • AWS::EC2::VPCEndpoint 	You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.								
Public subnets	<ul style="list-style-type: none"> • AWS::EC2::Subnet • AWS::EC2::SubnetNetworkACLAssociation 	Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules.								
Internet gateway	<ul style="list-style-type: none"> • AWS::EC2::InternetGateway • AWS::EC2::VPCGatewayAttachment • AWS::EC2::RouteTable • AWS::EC2::Route • AWS::EC2::SubnetRouteTableAssociation • AWS::EC2::NatGateway • AWS::EC2::EIP 	You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios.								
Network access control	<ul style="list-style-type: none"> • AWS::EC2::NetworkACL • AWS::EC2::NetworkACLEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th> <th>Reason</th> </tr> </thead> <tbody> <tr> <td>80</td> <td>Inbound HTTP traffic</td> </tr> <tr> <td>443</td> <td>Inbound HTTPS traffic</td> </tr> <tr> <td>22</td> <td>Inbound SSH traffic</td> </tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic	443	Inbound HTTPS traffic	22	Inbound SSH traffic
Port	Reason									
80	Inbound HTTP traffic									
443	Inbound HTTPS traffic									
22	Inbound SSH traffic									

Component	AWS type	Description	
		1024 - 65535	Inbound ephemeral traffic
		0 - 65535	Outbound ephemeral traffic
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them.	

2.6.3.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

2.6.3.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, Internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

2.6.3.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

2.6.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

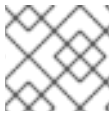
If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

2.6.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

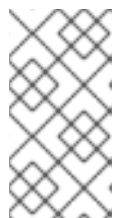
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

Identity added: /home/<you>/<path>/<file_name> (<computer_name>)

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

2.6.6. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your bastion host.
- Have the **imageContentSources** values that were generated during mirror registry creation.
- Obtain the contents of the certificate for your mirror registry.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.
 - a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}'
```

For **<bastion_host_name>**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

- b. Add the **additionalTrustBundle** parameter and value.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  /-----END CERTIFICATE-----
```

The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

- c. Define the subnets for the VPC to install the cluster in:

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

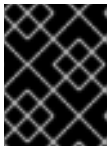
- d. Add the image content resources, which look like this excerpt:

```
imageContentSources:
```


- mirrors:
 - <bastion_host_name>:5000/<repo_name>/release
 - source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
 - <bastion_host_name>:5000/<repo_name>/release
 - source: registry.example.com/ocp/release

To complete these values, use the **imageContentSources** that you recorded during mirror registry creation.

3. Make any other modifications to the **install-config.yaml** file that you require. You can find more information about the available parameters in the **Installation configuration parameters** section.
4. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

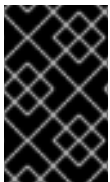
2.6.6.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

2.6.6.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 2.10. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String


Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


2.6.6.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 2.11. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .

Parameter	Description	Values
networking.serviceNetwork	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	<p>Required if you use networking.machineNetwork. An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24.</p>	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div>


2.6.6.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 2.12. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <p> NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <p> NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

2.6.6.1.4. Optional AWS configuration parameters

Optional AWS configuration parameters are described in the following table:

Table 2.13. Optional AWS parameters

Parameter	Description	Values
<code>compute.platform.aws.rootVolume.iops</code>	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
<code>compute.platform.aws.rootVolume.size</code>	The size in GiB of the root volume.	Integer, for example 500 .
<code>compute.platform.aws.rootVolume.type</code>	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
<code>compute.platform.aws.type</code>	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .
<code>compute.platform.aws.zones</code>	The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
<code>compute.aws.region</code>	The AWS region that the installation program creates compute resources in.	Any valid AWS region , such as us-east-1 .
<code>controlPlane.platform.aws.type</code>	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
<code>controlPlane.platform.aws.zones</code>	The availability zones where the installation program creates machines for the control plane machine pool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
<code>controlPlane.aws.region</code>	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
<code>platform.aws.userTags</code>	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.

Parameter	Description	Values
platform.aws.subnets	If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone.	Valid subnet IDs.

2.6.6.2. Sample customized `install-config.yaml` file for AWS

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com ①
controlPlane: ②
  hyperthreading: Enabled ③ ④
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
    rootVolume:
      iops: 4000
      size: 500
      type: io1
      type: m5.xlarge ⑤
    replicas: 3
  compute: ⑥
  - hyperthreading: Enabled ⑦
    name: worker
    platform:
      aws:
        rootVolume:
          iops: 2000

```

```

    size: 500
    type: io1 8
    type: c5.4xlarge
    zones:
    - us-west-2c
  replicas: 3
  metadata:
    name: test-cluster 9
  networking:
    clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
    machineNetwork:
    - cidr: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
    - 172.30.0.0/16
  platform:
    aws:
      region: us-west-2 10
      userTags:
        adminContact: jdoe
        costCenter: 7536
      subnets: 11
      - subnet-1
      - subnet-2
      - subnet-3
    fips: false 12
    sshKey: ssh-ed25519 AAAA... 13
    pullSecret: '{"auths":{"<local_registry>":{"auth": "<credentials>","email": "you@example.com"}}}' 14
    additionalTrustBundle: | 15
      -----BEGIN CERTIFICATE-----
      <MY_TRUSTED_CA_CERT>
      -----END CERTIFICATE-----
    imageContentSources: 16
    - mirrors:
      - <local_registry>/<local_repository_name>/release
      source: quay.io/openshift-release-dev/ocp-release
    - mirrors:
      - <local_registry>/<local_repository_name>/release
      source: registry.svc.ci.openshift.org/ocp/release

```

1 9 10 Required. The installation program prompts you for this value.

2 6 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 5 Whether to enable or disable simultaneous multithreading, or **hypertreading**. By default,



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

- 8 To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 11 If you provide your own VPC, specify subnets for each availability zone that your cluster uses.
- 12 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 13 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 14 For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 15 Provide the contents of the certificate file that you used for your mirror registry.
- 16 Provide the **imageContentSources** section from the output of the command to mirror the repository.

2.6.6.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

2.6.7. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

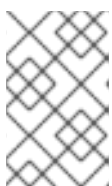
- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

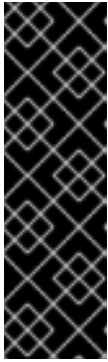
```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1** For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

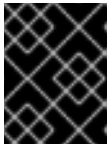
**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

- Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

2.6.8. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.6.8.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

- Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
- Select your infrastructure provider, and, if applicable, your installation type.
- In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
- Unpack the archive:

```
$ tar xvzf <file>
```

- Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.6.8.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.6.8.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.6.9. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

2.6.10. Next steps

- [Customize your cluster](#).
- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to [use Operator Lifecycle Manager \(OLM\) on restricted networks](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).

2.7. INSTALLING A CLUSTER ON AWS INTO AN EXISTING VPC

In OpenShift Container Platform version 4.5, you can install a cluster into an existing Amazon Virtual Private Cloud (VPC) on Amazon Web Services (AWS). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

2.7.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.

- [Configure an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.7.2. About using a custom VPC

In OpenShift Container Platform 4.5, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

2.7.2.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VPC options like DHCP, so you must do so before you install the cluster.

Your VPC must meet the following characteristics:

- The VPC's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VPC must not use the **kubernetes.io/cluster/.*: owned** tag.
- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See [DNS Support in Your VPC](#) in the AWS documentation.

If you use a cluster with public access, you must create a public and a private subnet for each availability zone that your cluster uses.

The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. Review the current [Tag Restrictions](#) in the AWS documentation to ensure that the installation program can add a tag to each subnet that you specify.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

Component	AWS type	Description
VPC	<ul style="list-style-type: none"> • AWS::EC2::VPC • AWS::EC2::VPCEndpoint 	You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.
Public subnets	<ul style="list-style-type: none"> • AWS::EC2::Subnet • AWS::EC2::SubnetNetworkACLAssociation 	Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules.

Component	AWS type	Description												
Internet gateway	<ul style="list-style-type: none"> ● AWS::EC2::InternetGateway ● AWS::EC2::VPCGatewayAttachment ● AWS::EC2::RouteTable ● AWS::EC2::Route ● AWS::EC2::SubnetRouteTableAssociation ● AWS::EC2::NatGateway ● AWS::EC2::EIP 	<p>You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios.</p>												
Network access control	<ul style="list-style-type: none"> ● AWS::EC2::NetworkAcl ● AWS::EC2::NetworkAclEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th> <th>Reason</th> </tr> </thead> <tbody> <tr> <td>80</td> <td>Inbound HTTP traffic</td> </tr> <tr> <td>443</td> <td>Inbound HTTPS traffic</td> </tr> <tr> <td>22</td> <td>Inbound SSH traffic</td> </tr> <tr> <td>1024 - 65535</td> <td>Inbound ephemeral traffic</td> </tr> <tr> <td>0 - 65535</td> <td>Outbound ephemeral traffic</td> </tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic	443	Inbound HTTPS traffic	22	Inbound SSH traffic	1024 - 65535	Inbound ephemeral traffic	0 - 65535	Outbound ephemeral traffic
Port	Reason													
80	Inbound HTTP traffic													
443	Inbound HTTPS traffic													
22	Inbound SSH traffic													
1024 - 65535	Inbound ephemeral traffic													
0 - 65535	Outbound ephemeral traffic													
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	<p>Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them.</p>												

2.7.2.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

2.7.2.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, Internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

2.7.2.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

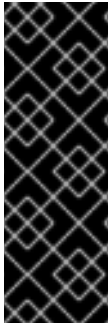
2.7.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

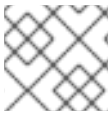


IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.7.4. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

2.7.5. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

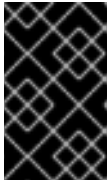
Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

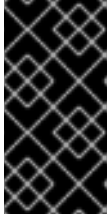
1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.7.6. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

**IMPORTANT**

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **AWS** as the platform to target.
 - iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

2.7.6.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

2.7.6.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 2.14. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .


Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

2.7.6.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 2.15. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	<p>Object</p>  <p>NOTE</p> <p>You cannot modify parameters specified by the networking object after installation.</p>

Parameter	Description	Values
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.



2.7.6.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 2.16. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

2.7.6.1.4. Optional AWS configuration parameters

Optional AWS configuration parameters are described in the following table:

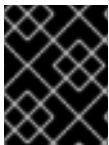
Table 2.17. Optional AWS parameters

Parameter	Description	Values
<code>compute.platform.aws.rootVolume.iops</code>	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
<code>compute.platform.aws.rootVolume.size</code>	The size in GiB of the root volume.	Integer, for example 500 .
<code>compute.platform.aws.rootVolume.type</code>	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
<code>compute.platform.aws.type</code>	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .
<code>compute.platform.aws.zones</code>	The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
<code>compute.aws.region</code>	The AWS region that the installation program creates compute resources in.	Any valid AWS region , such as us-east-1 .
<code>controlPlane.platform.aws.type</code>	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
<code>controlPlane.platform.aws.zones</code>	The availability zones where the installation program creates machines for the control plane machine pool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
<code>controlPlane.aws.region</code>	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
<code>platform.aws.userTags</code>	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.

Parameter	Description	Values
platform.aws.subnets	If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone.	Valid subnet IDs.

2.7.6.2. Sample customized `install-config.yaml` file for AWS

You can customize the **`install-config.yaml`** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **`install-config.yaml`** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com ①
controlPlane: ②
  hyperthreading: Enabled ③ ④
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
    rootVolume:
      iops: 4000
      size: 500
      type: io1
      type: m5.xlarge ⑤
    replicas: 3
  compute: ⑥
  - hyperthreading: Enabled ⑦
    name: worker
    platform:
      aws:
        rootVolume:
          iops: 2000
          size: 500

```

```

    type: io1 8
    type: c5.4xlarge
    zones:
    - us-west-2c
    replicas: 3
  metadata:
    name: test-cluster 9
  networking:
    clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
    machineNetwork:
    - cidr: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
    - 172.30.0.0/16
  platform:
    aws:
      region: us-west-2 10
      userTags:
        adminContact: jdoe
        costCenter: 7536
      subnets: 11
      - subnet-1
      - subnet-2
      - subnet-3
    fips: false 12
    sshKey: ssh-ed25519 AAAA... 13
    pullSecret: '{"auths": ...}' 14

```

1 9 10 14 Required. The installation program prompts you for this value.

2 6 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

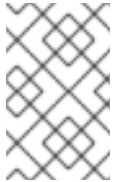
4 5 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

- 8 To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 11 If you provide your own VPC, specify subnets for each availability zone that your cluster uses.
- 12 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 13 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

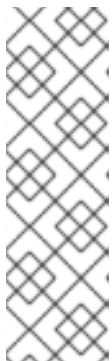
For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

2.7.6.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
```

```

additionalTrustBundle: | 4
-----BEGIN CERTIFICATE-----
<MY_TRUSTED_CA_CERT>
-----END CERTIFICATE-----
...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.



NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

2.7.7. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

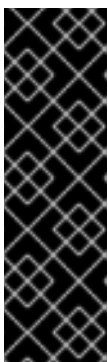
- 1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

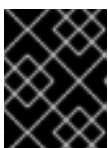
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

2.7.8. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.7.8.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.7.8.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

-

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.7.8.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.7.9. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

2.7.10. Next steps

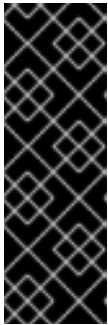
- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- If necessary, you can [remove cloud provider credentials](#).

2.8. INSTALLING A PRIVATE CLUSTER ON AWS

In OpenShift Container Platform version 4.5, you can install a private cluster into an existing VPC on Amazon Web Services (AWS). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

2.8.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#) . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.8.2. Private clusters

If your environment does not require an external Internet connection, you can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the Internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your

cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

To deploy a private cluster, you must use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.

Additionally, you must deploy a private cluster from a machine that has access the API services for the cloud you provision to, the hosts on the network that you provision, and to the internet to obtain installation media. You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

2.8.2.1. Private clusters in AWS

To create a private cluster on Amazon Web Services (AWS), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for access from only the private network.

The cluster still requires access to Internet to access the AWS APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public load balancers, which support public ingress
- A public Route 53 zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private Route 53 zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

2.8.2.1.1. Limitations

The ability to add public functionality to a private cluster is limited.

- You cannot make the Kubernetes API endpoints public after installation without taking additional actions, including creating public subnets in the VPC for each availability zone in use, creating a public load balancer, and configuring the control plane security groups to allow traffic from Internet on 6443 (Kubernetes API port).
- If you use a public Service type load balancer, you must tag a public subnet in each availability zone with **kubernetes.io/cluster/<cluster-infra-id>: shared** so that AWS can use them to create public load balancers.

2.8.3. About using a custom VPC

In OpenShift Container Platform 4.5, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

2.8.3.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VPC options like DHCP, so you must do so before you install the cluster.

Your VPC must meet the following characteristics:

- The VPC's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VPC must not use the **kubernetes.io/cluster/.*: owned** tag.
- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See [DNS Support in Your VPC](#) in the AWS documentation.

If you use a cluster with public access, you must create a public and a private subnet for each availability zone that your cluster uses.

The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. Review the current [Tag Restrictions](#) in the AWS documentation to ensure that the installation program can add a tag to each subnet that you specify.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

Component	AWS type	Description												
VPC	<ul style="list-style-type: none"> ● AWS::EC2::VPC ● AWS::EC2::VPCEndpoint 	You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.												
Public subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::SubnetNetworkACLAssociation 	Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules.												
Internet gateway	<ul style="list-style-type: none"> ● AWS::EC2::InternetGateway ● AWS::EC2::VPCGatewayAttachment ● AWS::EC2::RouteTable ● AWS::EC2::Route ● AWS::EC2::SubnetRouteTableAssociation ● AWS::EC2::NatGateway ● AWS::EC2::EIP 	You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios.												
Network access control	<ul style="list-style-type: none"> ● AWS::EC2::NetworkACL ● AWS::EC2::NetworkACLEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th> <th>Reason</th> </tr> </thead> <tbody> <tr> <td>80</td> <td>Inbound HTTP traffic</td> </tr> <tr> <td>443</td> <td>Inbound HTTPS traffic</td> </tr> <tr> <td>22</td> <td>Inbound SSH traffic</td> </tr> <tr> <td>1024 - 65535</td> <td>Inbound ephemeral traffic</td> </tr> <tr> <td>0 - 65535</td> <td>Outbound ephemeral traffic</td> </tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic	443	Inbound HTTPS traffic	22	Inbound SSH traffic	1024 - 65535	Inbound ephemeral traffic	0 - 65535	Outbound ephemeral traffic
Port	Reason													
80	Inbound HTTP traffic													
443	Inbound HTTPS traffic													
22	Inbound SSH traffic													
1024 - 65535	Inbound ephemeral traffic													
0 - 65535	Outbound ephemeral traffic													

Component	AWS type	Description
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them.

2.8.3.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

2.8.3.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, Internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

2.8.3.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

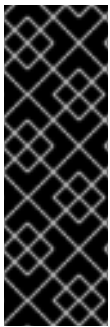
2.8.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.8.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

2.8.6. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

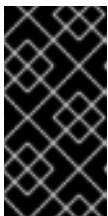
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.8.7. Manually creating the installation configuration file

For installations of a private OpenShift Container Platform cluster that are only accessible from an internal network and are not visible to the Internet, you must manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

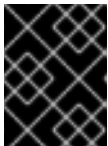
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

2.8.7.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

2.8.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 2.18. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object


Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


2.8.7.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 2.19. Network parameters


Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


2.8.7.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 2.20. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 2;"> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

2.8.7.1.4. Optional AWS configuration parameters

Optional AWS configuration parameters are described in the following table:

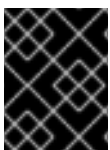
Table 2.21. Optional AWS parameters

Parameter	Description	Values
compute.platform.aws.rootVolume.iops	The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.	Integer, for example 4000 .
compute.platform.aws.rootVolume.size	The size in GiB of the root volume.	Integer, for example 500 .
compute.platform.aws.rootVolume.type	The instance type of the root volume.	Valid AWS EBS instance type , such as io1 .
compute.platform.aws.type	The EC2 instance type for the compute machines.	Valid AWS instance type , such as c5.9xlarge .
compute.platform.aws.zones	The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
compute.aws.region	The AWS region that the installation program creates compute resources in.	Any valid AWS region , such as us-east-1 .

Parameter	Description	Values
controlPlane.platform.aws.type	The EC2 instance type for the control plane machines.	Valid AWS instance type , such as c5.9xlarge .
controlPlane.platform.aws.zones	The availability zones where the installation program creates machines for the control plane machine pool.	A list of valid AWS availability zones, such as us-east-1c , in a YAML sequence .
controlPlane.aws.region	The AWS region that the installation program creates control plane resources in.	Valid AWS region , such as us-east-1 .
platform.aws.userTags	A map of keys and values that the installation program adds as tags to all resources that it creates.	Any valid YAML map, such as key value pairs in the <key>: <value> format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.
platform.aws.subnets	If you provide the VPC instead of allowing the installation program to create the VPC for you, specify the subnet for the cluster to use. The subnet must be part of the same machineNetwork[].cidr ranges that you specify. For a standard cluster, specify a public and a private subnet for each availability zone. For a private cluster, specify a private subnet for each availability zone.	Valid subnet IDs.

2.8.7.2. Sample customized `install-config.yaml` file for AWS

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

apiVersion: v1

baseDomain: example.com **1**


```

controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    aws:
      zones:
        - us-west-2a
        - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
        type: m5.xlarge 5
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
        type: c5.4xlarge
      zones:
        - us-west-2c
      replicas: 3
  metadata:
    name: test-cluster 9
  networking:
    clusterNetwork:
      - cidr: 10.128.0.0/14
        hostPrefix: 23
    machineNetwork:
      - cidr: 10.0.0.0/16
    networkType: OpenShiftSDN
    serviceNetwork:
      - 172.30.0.0/16
  platform:
    aws:
      region: us-west-2 10
      userTags:
        adminContact: jdoe
        costCenter: 7536
      subnets: 11
        - subnet-1
        - subnet-2
        - subnet-3
  fips: false 12
  sshKey: ssh-ed25519 AAAA... 13
  publish: Internal 14
  pullSecret: '{"auths": ...}' 15

```

1 9 10 15 Required. The installation program prompts you for this value.

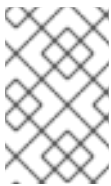
- 2 6 If you do not provide these parameters and values, the installation program provides the default value.
- 3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 4 5 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger instance types, such as **m4.2xlarge** or **m5.2xlarge**, for your machines if you disable simultaneous multithreading.

- 8 To configure faster storage for etcd, especially for larger clusters, set the storage type as **io1** and set **iops** to **2000**.
- 11 If you provide your own VPC, specify subnets for each availability zone that your cluster uses.
- 12 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 13 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 14 How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the Internet. The default value is **External**.

2.8.7.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.

- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

- Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

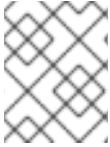
- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

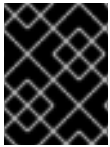
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

2.8.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

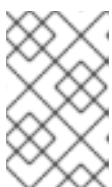
Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

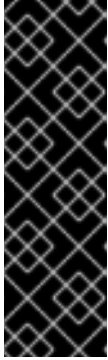
1 For **<installation_directory>**, specify the

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

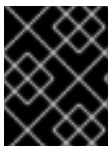
The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

2.8.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.8.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.8.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.8.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.8.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

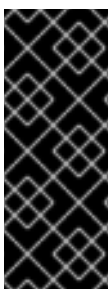
2.8.11. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- If necessary, you can [remove cloud provider credentials](#).

2.9. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN AWS BY USING CLOUDFORMATION TEMPLATES

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) that uses infrastructure that you provide.

One way to create this infrastructure is to use the provided CloudFormation templates. You can modify the templates to customize your infrastructure or use the information that they contain to create AWS objects according to your company's policies.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several CloudFormation templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

2.9.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.



IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#) in the AWS documentation.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.9.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.9.3. Required AWS infrastructure components

To install OpenShift Container Platform on user-provisioned infrastructure in Amazon Web Services (AWS), you must manually create both the machines and their supporting infrastructure.

For more information about the integration testing for different platforms, see the [OpenShift Container Platform 4.x Tested Integrations](#) page.

You can use the provided Cloud Formation templates to create this infrastructure, you can manually create the components, or you can reuse existing infrastructure that meets the cluster requirements. Review the Cloud Formation templates for more details about how the components interrelate.

2.9.3.1. Cluster machines

You need **AWS::EC2::Instance** objects for the following machines:

- A bootstrap machine. This machine is required during installation, but you can remove it after your cluster deploys.
- Three control plane machines. The control plane machines are not governed by a machine set.
- Compute machines. You must create at least two compute machines, which are also known as worker machines, during installation. These machines are not governed by a machine set.

You can use the following instance types for the cluster machines with the provided Cloud Formation templates.



IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

Table 2.22. Instance types for machines

Instance type	Bootstrap	Control plane	Compute
i3.large	x		
m4.large or m5.large			x
m4.xlarge or m5.xlarge		x	x

Instance type	Bootstrap	Control plane	Compute
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.8xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
c4.large			x
c4.xlarge			x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x

You might be able to use other instance types that meet the specifications of these instance types.

2.9.3.2. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

2.9.3.3. Other infrastructure components

- A VPC
- DNS entries
- Load balancers (classic or network) and listeners
- A public and a private Route 53 zone
- Security groups
- IAM roles
- S3 buckets

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

Component	AWS type	Description
VPC	<ul style="list-style-type: none"> • AWS::EC2::VPC • AWS::EC2::VPCEndpoint 	You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.
Public subnets	<ul style="list-style-type: none"> • AWS::EC2::Subnet • AWS::EC2::SubnetNetworkACLAssociation 	Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules.

Component	AWS type	Description												
Internet gateway	<ul style="list-style-type: none"> ● AWS::EC2::InternetGateway ● AWS::EC2::VPCElasticGatewayAttachment ● AWS::EC2::RouteTable ● AWS::EC2::Route ● AWS::EC2::SubnetRouteTableAssociation ● AWS::EC2::NatGateway ● AWS::EC2::EIP 	<p>You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios.</p>												
Network access control	<ul style="list-style-type: none"> ● AWS::EC2::NetworkAcl ● AWS::EC2::NetworkAclEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th> <th>Reason</th> </tr> </thead> <tbody> <tr> <td>80</td> <td>Inbound HTTP traffic</td> </tr> <tr> <td>443</td> <td>Inbound HTTPS traffic</td> </tr> <tr> <td>22</td> <td>Inbound SSH traffic</td> </tr> <tr> <td>1024 - 65535</td> <td>Inbound ephemeral traffic</td> </tr> <tr> <td>0 - 65535</td> <td>Outbound ephemeral traffic</td> </tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic	443	Inbound HTTPS traffic	22	Inbound SSH traffic	1024 - 65535	Inbound ephemeral traffic	0 - 65535	Outbound ephemeral traffic
Port	Reason													
80	Inbound HTTP traffic													
443	Inbound HTTPS traffic													
22	Inbound SSH traffic													
1024 - 65535	Inbound ephemeral traffic													
0 - 65535	Outbound ephemeral traffic													
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	<p>Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them.</p>												

Required DNS and load balancing components

Your DNS and load balancer configuration needs to use a public hosted zone and can use a private hosted zone similar to the one that the installation program uses if it provisions the cluster's infrastructure. You must create a DNS entry that resolves to your load balancer. An entry for **api**.

`<cluster_name>.<domain>` must point to the external load balancer, and an entry for `api-int.<cluster_name>.<domain>` must point to the internal load balancer.

The cluster also requires load balancers and listeners for port 6443, which are required for the Kubernetes API and its extensions, and port 22623, which are required for the Ignition config files for new machines. The targets will be the master nodes. Port 6443 must be accessible to both clients external to the cluster and nodes within the cluster. Port 22623 must be accessible to nodes within the cluster.

Component	AWS type	Description
DNS	AWS::Route53::HostedZone	The hosted zone for your internal DNS.
etcd record sets	AWS::Route53::RecordSet	The registration records for etcd for your control plane machines.
Public load balancer	AWS::ElasticLoadBalancingV2::LoadBalancer	The load balancer for your public subnets.
External API server record	AWS::Route53::RecordSetGroup	Alias records for the external API server.
External listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 6443 for the external load balancer.
External target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the external load balancer.
Private load balancer	AWS::ElasticLoadBalancingV2::LoadBalancer	The load balancer for your private subnets.
Internal API server record	AWS::Route53::RecordSetGroup	Alias records for the internal API server.
Internal listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 22623 for the internal load balancer.

Component	AWS type	Description
Internal target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the internal load balancer.
Internal listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 6443 for the internal load balancer.
Internal target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the internal load balancer.

Security groups

The control plane and worker machines require access to the following ports:

Group	Type	IP Protocol	Port range
MasterSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
		tcp	6443
		tcp	22623
WorkerSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
BootstrapSecurityGroup	AWS::EC2::SecurityGroup	tcp	22
		tcp	19531

Control plane Ingress

The control plane machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

Ingress group	Description	IP protocol	Port range
MasterIngressEtcd	etcd	tcp	2379- 2380

Ingress group	Description	IP protocol	Port range
MasterIngress Vxlan	Vxlan packets	udp	4789
MasterIngress WorkerVxlan	Vxlan packets	udp	4789
MasterIngress Internal	Internal cluster communication and Kubernetes proxy metrics	tcp	9000 - 9999
MasterIngress WorkerInternal	Internal cluster communication	tcp	9000 - 9999
MasterIngress Kube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress WorkerKube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
MasterIngress WorkerIngressServices	Kubernetes Ingress services	tcp	30000 - 32767

Worker Ingress

The worker machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

Ingress group	Description	IP protocol	Port range
WorkerIngress Vxlan	Vxlan packets	udp	4789
WorkerIngress WorkerVxlan	Vxlan packets	udp	4789
WorkerIngress Internal	Internal cluster communication	tcp	9000 - 9999
WorkerIngress WorkerInternal	Internal cluster communication	tcp	9000 - 9999

Ingress group	Description	IP protocol	Port range
WorkerIngress Kube	Kubernetes kubelet, scheduler, and controller manager	tcp	10250
WorkerIngress WorkerKube	Kubernetes kubelet, scheduler, and controller manager	tcp	10250
WorkerIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
WorkerIngress WorkerIngress Services	Kubernetes Ingress services	tcp	30000 - 32767

Roles and instance profiles

You must grant the machines permissions in AWS. The provided CloudFormation templates grant the machines permission the following **AWS::IAM::Role** objects and provide a **AWS::IAM::InstanceProfile** for each set of roles. If you do not use the templates, you can grant the machines the following broad permissions or the following individual permissions.

Role	Effect	Action	Resource
Master	Allow	ec2:*	*
	Allow	elasticloadbalancing:*	*
	Allow	iam:PassRole	*
	Allow	s3:GetObject	*
Worker	Allow	ec2:Describe*	*
Bootstrap	Allow	ec2:Describe*	*
	Allow	ec2:AttachVolume	*
	Allow	ec2:DetachVolume	*

2.9.3.4. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 2.11. Required EC2 permissions for installation

- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2>CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2>DeleteSecurityGroup**
- **ec2>DeleteSnapshot**
- **ec2>DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**

- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 2.12. Required permissions for creating network resources during installation

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**

- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



NOTE

If you use an existing VPC, your account does not require these permissions for creating network resources.

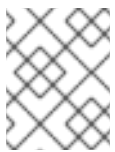
Example 2.13. Required Elastic Load Balancing permissions for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing>CreateListener**
- **elasticloadbalancing>CreateLoadBalancer**
- **elasticloadbalancing>CreateLoadBalancerListeners**
- **elasticloadbalancing>CreateTargetGroup**
- **elasticloadbalancing>DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**

- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

Example 2.14. Required IAM permissions for installation

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam>DeleteRole**
- **iam>DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



NOTE

If you have not created an elastic load balancer (ELB) in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 2.15. Required Route 53 permissions for installation

- **route53:ChangeResourceRecordSets**

- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53>DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Example 2.16. Required S3 permissions for installation

- **s3:CreateBucket**
- **s3>DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**

- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

Example 2.17. S3 permissions that cluster Operators require

- **s3>DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Example 2.18. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2>DeleteNetworkInterface**
- **ec2>DeleteVolume**
- **elasticloadbalancing>DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3>DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 2.19. Required permissions to delete network resources

- **ec2>DeleteDhcpOptions**

- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**



NOTE

If you use an existing VPC, your account does not require these permissions to delete network resources.

Example 2.20. Additional IAM and S3 permissions that are required to create manifests

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam>DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam>ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam>ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**

- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

2.9.4. Obtaining the installation program

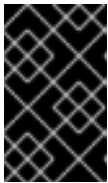
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

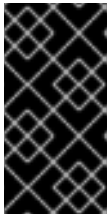
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

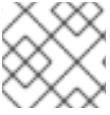
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

2.9.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

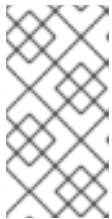
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

Identity added: /home/<you>/<path>/<file_name> (<computer_name>)

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

2.9.6. Creating the installation files for AWS

To install OpenShift Container Platform on Amazon Web Services (AWS) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files.

2.9.6.1. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Obtain the **install-config.yaml** file.
 - a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

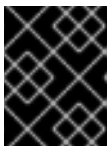
**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **aws** as the platform to target.
 - iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
```

3. Optional: Back up the **install-config.yaml** file.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

2.9.6.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

2.9.6.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:

- Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
- Locate the `mastersSchedulable` parameter and set its value to **False**.
- Save and exit the file.

- Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}
```

- ❶ Remove this section completely.
- ❷ Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

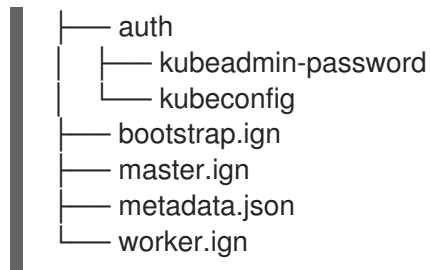
- Obtain the Ignition config files:

```
$. /openshift-install create ignition-configs --dir=<installation_directory> ❶
```

- ❶ For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:

```
.
```



2.9.7. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Amazon Web Services (AWS). The provided CloudFormation templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

2.9.8. Creating a VPC in AWS

You must create a VPC in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements, including VPN and route tables. The easiest way to create the VPC is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.

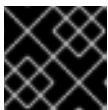
Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

- 1** The CIDR block for the VPC.
- 2** Specify a CIDR block in the format **x.x.x.x/16-24**.
- 3** The number of availability zones to deploy the VPC in.
- 4** Specify an integer between **1** and **3**.
- 5** The size of each subnet in each availability zone.
- 6** Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.

2. Copy the template from the **CloudFormation template for the VPC** section of this topic and save it as a YAML file on your computer. This template describes the VPC that your cluster requires.
3. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.

- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

VpcId	The ID of your VPC.
PublicSubnetIds	The IDs of the new public subnets.
PrivateSubnetIds	The IDs of the new private subnets.

2.9.8.1. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

Example 2.21. CloudFormation template for the VPC

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])|(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
```

Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"

Type: Number

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Network Configuration"

Parameters:

- VpcCidr

- SubnetBits

- Label:

default: "Availability Zones"

Parameters:

- AvailabilityZoneCount

ParameterLabels:

AvailabilityZoneCount:

default: "Availability Zone Count"

VpcCidr:

default: "VPC CIDR"

SubnetBits:

default: "Bits Per Subnet"

Conditions:

DoAz3: !Equals [3, !Ref AvailabilityZoneCount]

DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:

VPC:

Type: "AWS::EC2::VPC"

Properties:

EnableDnsSupport: "true"

EnableDnsHostnames: "true"

CidrBlock: !Ref VpcCidr

PublicSubnet:

Type: "AWS::EC2::Subnet"

Properties:

VpcId: !Ref VPC

CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

AvailabilityZone: !Select

- 0

- Fn::GetAZs: !Ref "AWS::Region"

PublicSubnet2:

Type: "AWS::EC2::Subnet"

Condition: DoAz2

Properties:

VpcId: !Ref VPC

CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

AvailabilityZone: !Select

- 1

- Fn::GetAZs: !Ref "AWS::Region"

PublicSubnet3:

Type: "AWS::EC2::Subnet"

Condition: DoAz3

Properties:

```

VpcId: !Ref VPC
CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
AvailabilityZone: !Select
- 2
- Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
- 0
- Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet

```

```
RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      Fn::GetAtt:
      - EIP
      - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      Fn::GetAtt:
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
```

```

Condition: DoAz2
Properties:
  Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP3
        - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:
  Type: "AWS::EC2::EIP"
  Condition: DoAz3
  Properties:
    Domain: vpc
Route3:
  Type: "AWS::EC2::Route"
  Condition: DoAz3
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable3
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:

```

```

    Ref: NAT3
S3Endpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal: '*'
          Action:
            - '*'
          Resource:
            - '*'
    RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
    ServiceName: !Join
      - "
      - - com.amazonaws
        - !Ref 'AWS::Region'
        - .s3
    VpId: !Ref VPC

Outputs:
  VpId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]

```

2.9.9. Creating networking and load balancing components in AWS

You must configure networking and load balancing (classic or network) in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. The easiest way to create these components is to modify the provided CloudFormation template, which also creates a hosted zone and subnet tags.

You can run the template multiple times within a single VPC.



NOTE

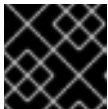
If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

Procedure

1. Obtain the Hosted Zone ID for the Route 53 zone that you specified in the **install-config.yaml** file for your cluster. You can obtain this ID from the AWS console or by running the following command:



IMPORTANT

You must enter the command on a single line.

```
$ aws route53 list-hosted-zones-by-name |
jq --arg name "<route53_domain>." \ 1
-r '.HostedZones | .[] | select(.Name=="($name)") | .Id'
```

- 1 For the **<route53_domain>**, specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

2. Create a JSON file that contains the parameter values that the template requires:

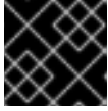
```
[
{
  "ParameterKey": "ClusterName", 1
  "ParameterValue": "mycluster" 2
},
{
  "ParameterKey": "InfrastructureName", 3
  "ParameterValue": "mycluster-<random_string>" 4
},
{
  "ParameterKey": "HostedZoneId", 5
  "ParameterValue": "<random_string>" 6
},
{
  "ParameterKey": "HostedZoneName", 7
  "ParameterValue": "example.com" 8
},
{
```

```

    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]

```

- 1 A short, representative cluster name to use for host names, etc.
 - 2 Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.
 - 3 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
 - 4 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
 - 5 The Route 53 public zone ID to register the targets with.
 - 6 Specify the Route 53 public zone ID, which as a format similar to **Z21IXYZABCZ2A4**. You can obtain this value from the AWS console.
 - 7 The Route 53 zone to register the targets with.
 - 8 Specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
 - 9 The public subnets that you created for your VPC.
 - 10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 11 The private subnets that you created for your VPC.
 - 12 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
 - 13 The VPC that you created for the cluster.
 - 14 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
3. Copy the template from the **CloudFormation template for the network and load balancers** section of this topic and save it as a YAML file on your computer. This template describes the networking and load balancing objects that your cluster requires.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
  --capabilities CAPABILITY_NAMED_IAM
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-dns**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

PrivateHostedZoneId	Hosted zone ID for the private DNS.
ExternalApiLoadBalancerName	Full name of the external API load balancer.
InternalApiLoadBalancerName	Full name of the internal API load balancer.
ApiServerDnsName	Full host name of the API server.
RegisterNlbTargetLambda	Lambda ARN useful to help register/deregister IP targets for these load balancers.
ExternalApiTargetGroupArn	ARN of external API target group.

InternalApiTargetGroupArn	ARN of internal API target group.
InternalServiceTargetGroupArn	ARN of internal service target group.

2.9.9.1. CloudFormation template for the network and load balancers

You can use the following CloudFormation template to deploy the networking objects and load balancers that you need for your OpenShift Container Platform cluster.

Example 2.22. CloudFormation template for the network and load balancers

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
    names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
    used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
    Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
    trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.

```

Type: List<AWS::EC2::Subnet::Id>
 VpcId:
 Description: The VPC-scoped resources will belong to this VPC.
 Type: AWS::EC2::VPC::Id

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:
 default: "Cluster Information"

Parameters:

- ClusterName
 - InfrastructureName

- Label:
 default: "Network Configuration"

Parameters:

- VpcId
 - PublicSubnets
 - PrivateSubnets

- Label:
 default: "DNS"

Parameters:

- HostedZoneName
 - HostedZoneId

ParameterLabels:

ClusterName:
 default: "Cluster Name"

InfrastructureName:
 default: "Infrastructure Name"

VpcId:
 default: "VPC ID"

PublicSubnets:
 default: "Public Subnets"

PrivateSubnets:
 default: "Private Subnets"

HostedZoneName:
 default: "Public Hosted Zone Name"

HostedZoneId:
 default: "Public Hosted Zone ID"

Resources:

ExtApiElb:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: !Join ["-", [!Ref InfrastructureName, "ext"]]

IpAddressType: ipv4

Subnets: !Ref PublicSubnets

Type: network

IntApiElb:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: !Join ["-", [!Ref InfrastructureName, "int"]]

Scheme: internal

IpAddressType: ipv4

Subnets: !Ref PrivateSubnets

Type: network

IntDns:

Type: "AWS::Route53::HostedZone"

Properties:

HostedZoneConfig:

Comment: "Managed by CloudFormation"

Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]

HostedZoneTags:

- Key: Name

Value: !Join ["-", [!Ref InfrastructureName, "int"]]

- Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]

Value: "owned"

VPCs:

- VPCId: !Ref VpcId

VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:

Type: AWS::Route53::RecordSetGroup

Properties:

Comment: Alias record for the API server

HostedZoneId: !Ref HostedZoneId

RecordSets:

- Name:

!Join [

".",

["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],

]

Type: A

AliasTarget:

HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID

DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:

Type: AWS::Route53::RecordSetGroup

Properties:

Comment: Alias record for the API server

HostedZoneId: !Ref IntDns

RecordSets:

- Name:

!Join [

".",

["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],

]

Type: A

AliasTarget:

HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID

DNSName: !GetAtt IntApiElb.DNSName

- Name:

!Join [

".",

["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],

]

Type: A

AliasTarget:

HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID

DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

DefaultActions:

- Type: forward

TargetGroupArn:

Ref: ExternalApiTargetGroup

LoadBalancerArn:

Ref: ExtApiElb

Port: 6443

Protocol: TCP

ExternalApiTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

HealthCheckIntervalSeconds: 10

HealthCheckPath: "/readyz"

HealthCheckPort: 6443

HealthCheckProtocol: HTTPS

HealthyThresholdCount: 2

UnhealthyThresholdCount: 2

Port: 6443

Protocol: TCP

TargetType: ip

VpcId:

Ref: VpcId

TargetGroupAttributes:

- Key: deregistration_delay.timeout_seconds

Value: 60

InternalApiListener:

Type: AWS::ElasticLoadBalancingV2::Listener

Properties:

DefaultActions:

- Type: forward

TargetGroupArn:

Ref: InternalApiTargetGroup

LoadBalancerArn:

Ref: IntApiElb

Port: 6443

Protocol: TCP

InternalApiTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

HealthCheckIntervalSeconds: 10

HealthCheckPath: "/readyz"

HealthCheckPort: 6443

HealthCheckProtocol: HTTPS

HealthyThresholdCount: 2

UnhealthyThresholdCount: 2

Port: 6443

Protocol: TCP

TargetType: ip

VpcId:
Ref: VpcId
TargetGroupAttributes:
- Key: deregistration_delay.timeout_seconds
Value: 60

InternalServiceInternalListener:
Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
DefaultActions:
- Type: forward
TargetGroupArn:
Ref: InternalServiceTargetGroup
LoadBalancerArn:
Ref: IntApiElb
Port: 22623
Protocol: TCP

InternalServiceTargetGroup:
Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
HealthCheckIntervalSeconds: 10
HealthCheckPath: "/healthz"
HealthCheckPort: 22623
HealthCheckProtocol: HTTPS
HealthyThresholdCount: 2
UnhealthyThresholdCount: 2
Port: 22623
Protocol: TCP
TargetType: ip
VpcId:
Ref: VpcId
TargetGroupAttributes:
- Key: deregistration_delay.timeout_seconds
Value: 60

RegisterTargetLambdalaRole:
Type: AWS::IAM::Role
Properties:
RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
AssumeRolePolicyDocument:
Version: "2012-10-17"
Statement:
- Effect: "Allow"
Principal:
Service:
- "lambda.amazonaws.com"
Action:
- "sts:AssumeRole"
Path: "/"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
PolicyDocument:
Version: "2012-10-17"
Statement:
- Effect: "Allow"

```

    Action:
      [
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets",
      ]
    Resource: !Ref InternalApiTargetGroup
  - Effect: "Allow"
  Action:
    [
      "elasticloadbalancing:RegisterTargets",
      "elasticloadbalancing:DeregisterTargets",
    ]
    Resource: !Ref InternalServiceTargetGroup
  - Effect: "Allow"
  Action:
    [
      "elasticloadbalancing:RegisterTargets",
      "elasticloadbalancing:DeregisterTargets",
    ]
    Resource: !Ref ExternalApiTargetGroup

RegisterNlbIpTargets:
  Type: "AWS::Lambda::Function"
  Properties:
    Handler: "index.handler"
    Role:
      Fn::GetAtt:
        - "RegisterTargetLambdalamRole"
        - "Arn"
    Code:
      ZipFile: |
import json
import boto3
import cfnresponse
def handler(event, context):
    elb = boto3.client('elbv2')
    if event['RequestType'] == 'Delete':
        elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
[TargetArn],Targets=[{'Id': event['ResourceProperties']['TargetIp']})
    elif event['RequestType'] == 'Create':
        elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']})
        responseData = {}
        cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
    Runtime: "python3.7"
    Timeout: 120

RegisterSubnetTagsLambdalamRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: "Allow"

```

```

Principal:
  Service:
    - "lambda.amazonaws.com"
  Action:
    - "sts:AssumeRole"
Path: "/"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
  PolicyDocument:
    Version: "2012-10-17"
    Statement:
    - Effect: "Allow"
      Action:
        [
          "ec2:DeleteTags",
          "ec2:CreateTags"
        ]
      Resource: "arn:aws:ec2:*:*:subnet/*"
    - Effect: "Allow"
      Action:
        [
          "ec2:DescribeSubnets",
          "ec2:DescribeTags"
        ]
      Resource: ""

```

RegisterSubnetTags:

Type: "AWS::Lambda::Function"

Properties:

Handler: "index.handler"

Role:

Fn::GetAtt:

- "RegisterSubnetTagsLambdalamRole"
- "Arn"

Code:

```

ZipFile: |
import json
import boto3
import cfnresponse
def handler(event, context):
    ec2_client = boto3.client('ec2')
    if event['RequestType'] == 'Delete':
        for subnet_id in event['ResourceProperties']['Subnets']:
            ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
    elif event['RequestType'] == 'Create':
        for subnet_id in event['ResourceProperties']['Subnets']:
            ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
        responseData = {}
        cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
Runtime: "python3.7"
Timeout: 120

```

RegisterPublicSubnetTags:

Type: Custom::SubnetRegister

Properties:

ServiceToken: !GetAtt RegisterSubnetTags.Arn

InfrastructureName: !Ref InfrastructureName

Subnets: !Ref PublicSubnets

RegisterPrivateSubnetTags:

Type: Custom::SubnetRegister

Properties:

ServiceToken: !GetAtt RegisterSubnetTags.Arn

InfrastructureName: !Ref InfrastructureName

Subnets: !Ref PrivateSubnets

Outputs:

PrivateHostedZoneId:

Description: Hosted zone ID for the private DNS, which is required for private records.

Value: !Ref IntDns

ExternalApiLoadBalancerName:

Description: Full name of the external API load balancer.

Value: !GetAtt ExtApiElb.LoadBalancerFullName

InternalApiLoadBalancerName:

Description: Full name of the internal API load balancer.

Value: !GetAtt IntApiElb.LoadBalancerFullName

ApiServerDnsName:

Description: Full hostname of the API server, which is required for the Ignition config files.

Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]

RegisterNlbIpTargetsLambda:

Description: Lambda ARN useful to help register or deregister IP targets for these load balancers.

Value: !GetAtt RegisterNlbIpTargets.Arn

ExternalApiTargetGroupArn:

Description: ARN of the external API target group.

Value: !Ref ExternalApiTargetGroup

InternalApiTargetGroupArn:

Description: ARN of the internal API target group.

Value: !Ref InternalApiTargetGroup

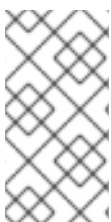
InternalServiceTargetGroupArn:

Description: ARN of the internal service target group.

Value: !Ref InternalServiceTargetGroup

2.9.10. Creating security group and roles in AWS

You must create security groups and roles in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. The easiest way to create these components is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

Procedure

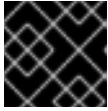
1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 The CIDR block for the VPC.
- 4 Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.
- 5 The private subnets that you created for your VPC.
- 6 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
- 7 The VPC that you created for the cluster.
- 8 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

2. Copy the template from the **CloudFormation template for security objects** section of this topic and save it as a YAML file on your computer. This template describes the security groups and roles that your cluster requires.

3. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml ❷
  --parameters file://<parameters>.json ❸
  --capabilities CAPABILITY_NAMED_IAM
```

- ❶ **<name>** is the name for the CloudFormation stack, such as **cluster-sec**. You need the name of this stack if you remove the cluster.
- ❷ **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- ❸ **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

MasterSecurityGroupID	Master Security Group ID
WorkerSecurityGroupID	Worker Security Group ID
MasterInstanceProfile	Master IAM Instance Profile
WorkerInstanceProfile	Worker IAM Instance Profile

2.9.10.1. CloudFormation template for security objects

You can use the following CloudFormation template to deploy the security objects that you need for your OpenShift Container Platform cluster.

Example 2.23. CloudFormation template for security objects

AWSTemplateFormatVersion: [2010-09-09](#)

Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:

InfrastructureName:

AllowedPattern: `^([a-zA-Z][a-zA-Z0-9\-_]{0,26})$`

MaxLength: [27](#)

MinLength: [1](#)

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of [27](#) characters.

Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.

Type: String

VpcCidr:

AllowedPattern: `^((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.)\.)\.)\{3\}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\V(1[6-9]|2[0-4]))$`

ConstraintDescription: CIDR block parameter must be in the form `x.x.x.x/16-24`.

Default: [10.0.0.0/16](#)

Description: CIDR block for VPC.

Type: String

VpcId:

Description: The VPC-scoped resources will belong to this VPC.

Type: `AWS::EC2::VPC::Id`

PrivateSubnets:

Description: The internal subnets.

Type: `List<AWS::EC2::Subnet::Id>`

Metadata:

`AWS::CloudFormation::Interface:`

ParameterGroups:

- Label:

default: `"Cluster Information"`

Parameters:

- InfrastructureName

- Label:

default: `"Network Configuration"`

Parameters:

- VpcId

- VpcCidr

- PrivateSubnets

ParameterLabels:

InfrastructureName:

default: `"Infrastructure Name"`

VpcId:

default: `"VPC ID"`

VpcCidr:

default: `"VPC CIDR"`

PrivateSubnets:

default: `"Private Subnets"`

Resources:

MasterSecurityGroup:

Type: `AWS::EC2::SecurityGroup`

Properties:

GroupDescription: Cluster Master Security Group

SecurityGroupIngress:

- IpProtocol: icmp
 - FromPort: 0
 - ToPort: 0
 - CidrIp: !Ref VpcCidr
- IpProtocol: tcp
 - FromPort: 22
 - ToPort: 22
 - CidrIp: !Ref VpcCidr
- IpProtocol: tcp
 - ToPort: 6443
 - FromPort: 6443
 - CidrIp: !Ref VpcCidr
- IpProtocol: tcp
 - FromPort: 22623
 - ToPort: 22623
 - CidrIp: !Ref VpcCidr

Vpclid: !Ref Vpclid

WorkerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Cluster Worker Security Group

SecurityGroupIngress:

- IpProtocol: icmp
 - FromPort: 0
 - ToPort: 0
 - CidrIp: !Ref VpcCidr
- IpProtocol: tcp
 - FromPort: 22
 - ToPort: 22
 - CidrIp: !Ref VpcCidr

Vpclid: !Ref Vpclid

MasterIngressEtcid:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: etcid
 FromPort: 2379
 ToPort: 2380
 IpProtocol: tcp

MasterIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Vxlan packets
 FromPort: 4789
 ToPort: 4789
 IpProtocol: udp

MasterIngressWorkerVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Vxlan packets
FromPort: 4789
ToPort: 4789
IpProtocol: udp

MasterIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

MasterIngressWorkerGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

MasterIngressInternal:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: tcp

MasterIngressWorkerInternal:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: tcp

MasterIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: udp

MasterIngressWorkerInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

MasterIngressKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes kubelet, scheduler and controller manager

FromPort: 10250

ToPort: 10259

IpProtocol: tcp

MasterIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

MasterIngressWorkerIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

MasterIngressIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services
FromPort: 30000
ToPort: 32767
IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Kubernetes ingress services
FromPort: 30000
ToPort: 32767
IpProtocol: udp

WorkerIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Vxlan packets
FromPort: 4789
ToPort: 4789
IpProtocol: udp

WorkerIngressMasterVxlan:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Vxlan packets
FromPort: 4789
ToPort: 4789
IpProtocol: udp

WorkerIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

WorkerIngressMasterGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

WorkerIngressInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

WorkerIngressMasterInternal:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: tcp

WorkerIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

WorkerIngressMasterInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal cluster communication

FromPort: 9000

ToPort: 9999

IpProtocol: udp

WorkerIngressKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes secure kubelet port

FromPort: 10250

ToPort: 10250

IpProtocol: tcp

WorkerIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal Kubernetes communication

FromPort: 10250

ToPort: 10250
IpProtocol: tcp

WorkerIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress
Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Kubernetes ingress services
 FromPort: 30000
 ToPort: 32767
 IpProtocol: tcp

WorkerIngressMasterIngressServices:

Type: AWS::EC2::SecurityGroupIngress
Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Kubernetes ingress services
 FromPort: 30000
 ToPort: 32767
 IpProtocol: tcp

WorkerIngressIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
 Description: Kubernetes ingress services
 FromPort: 30000
 ToPort: 32767
 IpProtocol: udp

WorkerIngressMasterIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
 GroupId: !GetAtt WorkerSecurityGroup.GroupId
 SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
 Description: Kubernetes ingress services
 FromPort: 30000
 ToPort: 32767
 IpProtocol: udp

MasterIamRole:

Type: AWS::IAM::Role
Properties:
 AssumeRolePolicyDocument:
 Version: "2012-10-17"
 Statement:
 - Effect: "Allow"
 Principal:
 Service:
 - "ec2.amazonaws.com"
 Action:
 - "sts:AssumeRole"
 Policies:

```

- PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
PolicyDocument:
  Version: "2012-10-17"
  Statement:
  - Effect: "Allow"
    Action:
    - "ec2:AttachVolume"
    - "ec2:AuthorizeSecurityGroupIngress"
    - "ec2:CreateSecurityGroup"
    - "ec2:CreateTags"
    - "ec2:CreateVolume"
    - "ec2>DeleteSecurityGroup"
    - "ec2>DeleteVolume"
    - "ec2:Describe*"
    - "ec2:DetachVolume"
    - "ec2:ModifyInstanceAttribute"
    - "ec2:ModifyVolume"
    - "ec2:RevokeSecurityGroupIngress"
    - "elasticloadbalancing:AddTags"
    - "elasticloadbalancing:AttachLoadBalancerToSubnets"
    - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
    - "elasticloadbalancing:CreateListener"
    - "elasticloadbalancing:CreateLoadBalancer"
    - "elasticloadbalancing:CreateLoadBalancerPolicy"
    - "elasticloadbalancing:CreateLoadBalancerListeners"
    - "elasticloadbalancing:CreateTargetGroup"
    - "elasticloadbalancing:ConfigureHealthCheck"
    - "elasticloadbalancing>DeleteListener"
    - "elasticloadbalancing>DeleteLoadBalancer"
    - "elasticloadbalancing>DeleteLoadBalancerListeners"
    - "elasticloadbalancing>DeleteTargetGroup"
    - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
    - "elasticloadbalancing:DeregisterTargets"
    - "elasticloadbalancing:Describe*"
    - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
    - "elasticloadbalancing:ModifyListener"
    - "elasticloadbalancing:ModifyLoadBalancerAttributes"
    - "elasticloadbalancing:ModifyTargetGroup"
    - "elasticloadbalancing:ModifyTargetGroupAttributes"
    - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
    - "elasticloadbalancing:RegisterTargets"
    - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
    - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
    - "kms:DescribeKey"
  Resource: "*"

```

```

MasterInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Roles:
    - Ref: "MasterIamRole"

```

```

WorkerIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:

```

```

Version: "2012-10-17"
Statement:
- Effect: "Allow"
Principal:
  Service:
    - "ec2.amazonaws.com"
Action:
- "sts:AssumeRole"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
  PolicyDocument:
    Version: "2012-10-17"
    Statement:
      - Effect: "Allow"
        Action:
          - "ec2:DescribeInstances"
          - "ec2:DescribeRegions"
        Resource: "*"

```

```

WorkerInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Roles:
      - Ref: "WorkerIamRole"

```

```

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

```

```

MasterInstanceProfile:
  Description: Master IAM Instance Profile
  Value: !Ref MasterInstanceProfile

```

```

WorkerInstanceProfile:
  Description: Worker IAM Instance Profile
  Value: !Ref WorkerInstanceProfile

```

2.9.11. RHCOS AMIs for the AWS infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) AMI for your Amazon Web Services (AWS) zone for your OpenShift Container Platform nodes.

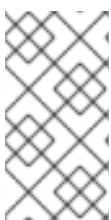
Table 2.23. RHCOS AMIs

AWS zone	AWS AMI
ap-northeast-1	ami-0530d04240177f118

AWS zone	AWS AMI
ap-northeast-2	ami-09e4cd700276785d2
ap-south-1	ami-0754b15d212830477
ap-southeast-1	ami-03b46cc4b1518c5a8
ap-southeast-2	ami-0a5b99ab2234a4e6a
ca-central-1	ami-012bc4ee3b6c673bc
eu-central-1	ami-02e08df1201f1c2f8
eu-north-1	ami-0309c9d2fadcb2d5a
eu-west-1	ami-0bdd69d8e7cd18188
eu-west-2	ami-0e610e967a62dbdfa
eu-west-3	ami-0e817e26f638a71ac
me-south-1	ami-024117d7c87b7ff08
sa-east-1	ami-08e62f746b94950c1
us-east-1	ami-077ede5bed2e431ea
us-east-2	ami-0f4ecf819275850dd
us-west-1	ami-0c4990e435bc6c5fe
us-west-2	ami-000d6e92357ac605c

2.9.12. Creating the bootstrap node in AWS

You must create the bootstrap node in Amazon Web Services (AWS) to use during OpenShift Container Platform cluster initialization. The easiest way to create this node is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your bootstrap node, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.

Procedure

1. Provide a location to serve the **bootstrap.ign** Ignition config file to your cluster. This file is located in your installation directory. One way to do this is to create an S3 bucket in your cluster's region and upload the Ignition config file to it.



IMPORTANT

The provided CloudFormation Template assumes that the Ignition config files for your cluster are served from an S3 bucket. If you choose to serve the files from another location, you must modify the templates.



NOTE

The bootstrap Ignition config file does contain secrets, like X.509 keys. The following steps provide basic security for the S3 bucket. To provide additional security, you can enable an S3 bucket policy to allow only certain users, such as the OpenShift IAM user, to access objects that the bucket contains. You can avoid S3 entirely and serve your bootstrap Ignition config file from any address that the bootstrap machine can reach.

- a. Create the bucket:

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** is the bucket name.

- b. Upload the **bootstrap.ign** Ignition config file to the bucket:

```
$ aws s3 cp bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign
```

- c. Verify that the file uploaded:

```
$ aws s3 ls s3://<cluster-name>-infra/
```

Example output

```
2019-04-03 16:15:16 314878 bootstrap.ign
```

2. Create a JSON file that contains the parameter values that the template requires:

```
[
```

```

{
  "ParameterKey": "InfrastructureName", 1
  "ParameterValue": "mycluster-<random_string>" 2
},
{
  "ParameterKey": "RhcOsAmi", 3
  "ParameterValue": "ami-<random_string>" 4
},
{
  "ParameterKey": "AllowedBootstrapSshCidr", 5
  "ParameterValue": "0.0.0.0/0" 6
},
{
  "ParameterKey": "PublicSubnet", 7
  "ParameterValue": "subnet-<random_string>" 8
},
{
  "ParameterKey": "MasterSecurityGroupID", 9
  "ParameterValue": "sg-<random_string>" 10
},
{
  "ParameterKey": "VpcID", 11
  "ParameterValue": "vpc-<random_string>" 12
},
{
  "ParameterKey": "BootstrapIgnitionLocation", 13
  "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
},
{
  "ParameterKey": "AutoRegisterELB", 15
  "ParameterValue": "yes" 16
},
{
  "ParameterKey": "RegisterNlbPTargetsLambdaArn", 17
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbPTargets-<random_string>" 18
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", 19
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
},
{
  "ParameterKey": "InternalApiTargetGroupArn", 21
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", 23
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
}
]

```

-
- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node.
- 4 Specify a valid **AWS::EC2::Image::Id** value.
- 5 CIDR block to allow SSH access to the bootstrap node.
- 6 Specify a CIDR block in the format **x.x.x.x/16-24**.
- 7 The public subnet that is associated with your VPC to launch the bootstrap node into.
- 8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
- 9 The master security group ID (for registering temporary rules)
- 10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
- 11 The VPC created resources will belong to.
- 12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
- 13 Location to fetch bootstrap Ignition config file from.
- 14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.
- 15 Whether or not to register a network load balancer (NLB).
- 16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
- 17 The ARN for NLB IP target registration lambda group.
- 18 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
- 19 The ARN for external API load balancer target group.
- 20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 21 The ARN for internal API load balancer target group.
- 22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 23 The ARN for internal service load balancer target group.
- 24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.

- Copy the template from the **CloudFormation template for the bootstrap machine** section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.
- Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> ❶
  --template-body file://<template>.yaml ❷
  --parameters file://<parameters>.json ❸
  --capabilities CAPABILITY_NAMED_IAM
```

- <name>** is the name for the CloudFormation stack, such as **cluster-bootstrap**. You need the name of this stack if you remove the cluster.
- <template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- <parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

Bootstrap InstanceId	The bootstrap Instance ID.
Bootstrap PublicIp	The bootstrap node public IP address.
Bootstrap PrivateIp	The bootstrap node private IP address.

2.9.12.1. CloudFormation template for the bootstrap machine

You can use the following CloudFormation template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster.

Example 2.24. CloudFormation template for the bootstrap machine

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)
```

Parameters:

InfrastructureName:

AllowedPattern: `^[a-zA-Z][a-zA-Z0-9\-\]{0,26}$`

MaxLength: `27`

MinLength: `1`

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of `27` characters.

Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.

Type: String

RhcOsAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: `AWS::EC2::Image::Id`

AllowedBootstrapSshCidr:

AllowedPattern: `^((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.)\.)\{3\}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\)(\{([0-9]|1[0-9]|2[0-9]|3[0-2])\})$`

ConstraintDescription: CIDR block parameter must be in the form `x.x.x.x/0-32`.

Default: `0.0.0.0/0`

Description: CIDR block to allow SSH access to the bootstrap node.

Type: String

PublicSubnet:

Description: The public subnet to launch the bootstrap node into.

Type: `AWS::EC2::Subnet::Id`

MasterSecurityGroupId:

Description: The master security group ID for registering temporary rules.

Type: `AWS::EC2::SecurityGroup::Id`

VpcId:

Description: The VPC-scoped resources will belong to this VPC.

Type: `AWS::EC2::VPC::Id`

BootstrapIgnitionLocation:

Default: `s3://my-s3-bucket/bootstrap.ign`

Description: Ignition config file location.

Type: String

AutoRegisterELB:

Default: `"yes"`

AllowedValues:

- `"yes"`

- `"no"`

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNlbIpTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group.

Type: String

Metadata:

`AWS::CloudFormation::Interface:`

`ParameterGroups:`

```

- Label:
  default: "Cluster Information"
Parameters:
- InfrastructureName
- Label:
  default: "Host Information"
Parameters:
- RhcosAmi
- BootstrapIgnitionLocation
- MasterSecurityGroupId
- Label:
  default: "Network Configuration"
Parameters:
- VpcId
- AllowedBootstrapSshCidr
- PublicSubnet
- Label:
  default: "Load Balancer Automation"
Parameters:
- AutoRegisterELB
- RegisterNlbPTargetsLambdaArn
- ExternalApiTargetGroupArn
- InternalApiTargetGroupArn
- InternalServiceTargetGroupArn
ParameterLabels:
InfrastructureName:
  default: "Infrastructure Name"
VpcId:
  default: "VPC ID"
AllowedBootstrapSshCidr:
  default: "Allowed SSH Source"
PublicSubnet:
  default: "Public Subnet"
RhcosAmi:
  default: "Red Hat Enterprise Linux CoreOS AMI ID"
BootstrapIgnitionLocation:
  default: "Bootstrap Ignition Source"
MasterSecurityGroupId:
  default: "Master Security Group ID"
AutoRegisterELB:
  default: "Use Provided ELB Automation"

```

Conditions:

```
DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
```

Resources:

```

BootstrapIamRole:
Type: AWS::IAM::Role
Properties:
AssumeRolePolicyDocument:
Version: "2012-10-17"
Statement:
- Effect: "Allow"
Principal:
Service:
- "ec2.amazonaws.com"

```

```
Action:
- "sts:AssumeRole"
Path: "/"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
  PolicyDocument:
    Version: "2012-10-17"
    Statement:
      - Effect: "Allow"
        Action: "ec2:Describe*"
        Resource: "*"
      - Effect: "Allow"
        Action: "ec2:AttachVolume"
        Resource: "*"
      - Effect: "Allow"
        Action: "ec2:DetachVolume"
        Resource: "*"
      - Effect: "Allow"
        Action: "s3:GetObject"
        Resource: "*"

BootstrapInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Path: "/"
    Roles:
      - Ref: "BootstrapIamRole"

BootstrapSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Bootstrap Security Group
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
    VpCid: !Ref VpCid

BootstrapInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    IamInstanceProfile: !Ref BootstrapInstanceProfile
    InstanceType: "i3.large"
    NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "BootstrapSecurityGroup"
          - !Ref "MasterSecurityGroup"
        SubnetId: !Ref "PublicSubnet"
```

```

UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"replace":{"source":"${S3Loc}","verification":{}}},"timeouts":
      {}, "version":"2.1.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
    - {
      S3Loc: !Ref BootstrapIgnitionLocation
    }

```

```

RegisterBootstrapApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

```

```

RegisterBootstrapInternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

```

```

RegisterBootstrapInternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

```

```

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

```

```

BootstrapPublicIp:
  Description: The bootstrap node public IP address.
  Value: !GetAtt BootstrapInstance.PublicIp

```

```

BootstrapPrivateIp:
  Description: The bootstrap node private IP address.
  Value: !GetAtt BootstrapInstance.PrivateIp

```

2.9.13. Creating the control plane machines in AWS

You must create the control plane machines in Amazon Web Services (AWS) for your cluster to use. The easiest way to create these nodes is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your control plane nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcocAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
]
```

```

{
  "ParameterKey": "Master2Subnet", 15
  "ParameterValue": "subnet-<random_string>" 16
},
{
  "ParameterKey": "MasterSecurityGroupID", 17
  "ParameterValue": "sg-<random_string>" 18
},
{
  "ParameterKey": "IgnitionLocation", 19
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
},
{
  "ParameterKey": "CertificateAuthorities", 21
  "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
},
{
  "ParameterKey": "MasterInstanceProfileName", 23
  "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
},
{
  "ParameterKey": "MasterInstanceType", 25
  "ParameterValue": "m4.xlarge" 26
},
{
  "ParameterKey": "AutoRegisterELB", 27
  "ParameterValue": "yes" 28
},
{
  "ParameterKey": "RegisterNlbTargetsLambdaArn", 29
  "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" 30
},
{
  "ParameterKey": "ExternalApiTargetGroupArn", 31
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
},
{
  "ParameterKey": "InternalApiTargetGroupArn", 33
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
},
{
  "ParameterKey": "InternalServiceTargetGroupArn", 35
  "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
}
]

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 Whether or not to perform DNS etcd registration.
- 6 Specify **yes** or **no**. If you specify **yes**, you must provide hosted zone information.
- 7 The Route 53 private zone ID to register the etcd targets with.
- 8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.
- 9 The Route 53 zone to register the targets with.
- 10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route 53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
- 11 13 15 A subnet, preferably private, to launch the control plane machines on.
- 12 14 16 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 17 The master security group ID to associate with master nodes.
- 18 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
- 19 The location to fetch control plane Ignition config file from.
- 20 Specify the generated Ignition config file location, https://api-int.<cluster_name>.<domain_name>:22623/config/master.
- 21 The base64 encoded certificate authority string to use.
- 22 Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 23 The IAM profile to associate with master nodes.
- 24 Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 25 The type of AWS instance to use for the control plane machines.
- 26 Allowed values:
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**

- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**

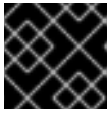


IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, specify an **m5** type, such as **m5.xlarge**, instead.

- 27 Whether or not to register a network load balancer (NLB).
 - 28 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
 - 29 The ARN for NLB IP target registration lambda group.
 - 30 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
 - 31 The ARN for external API load balancer target group.
 - 32 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 33 The ARN for internal API load balancer target group.
 - 34 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 35 The ARN for internal service load balancer target group.
 - 36 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
2. Copy the template from the **CloudFormation template for control plane machines** section of this topic and save it as a YAML file on your computer. This template describes the control plane machines that your cluster requires.

- If you specified an **m5** instance type as the value for **MasterInstanceType**, add that instance type to the **MasterInstanceType.AllowedValues** parameter in the CloudFormation template.
- Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-control-plane**. You need the name of this stack if you remove the cluster.
- 2** **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3** **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2.9.13.1. CloudFormation template for control plane machines

You can use the following CloudFormation template to deploy the control plane machines that you need for your OpenShift Container Platform cluster.

Example 2.25. CloudFormation template for control plane machines

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
      - "yes"
      - "no"
```

Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone information?

Type: String

PrivateHostedZoneId:

Description: The Route53 private zone ID to register the etcd targets with, such as Z21IXYZABCZ2A4.

Type: String

PrivateHostedZoneName:

Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit the trailing period.

Type: String

Master0Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master1Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master2Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

MasterSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: AWS::EC2::SecurityGroup::Id

IgnitionLocation:

Default: `https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master`

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: `data:text/plain;charset=utf-8;base64,ABC...xYz==`

Description: Base64 encoded certificate authority string to use.

Type: String

MasterInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

MasterInstanceType:

Default: `m4.xlarge`

Type: String

AllowedValues:

- "m4.xlarge"
- "m4.2xlarge"
- "m4.4xlarge"
- "m4.8xlarge"
- "m4.10xlarge"
- "m4.16xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"

- "no"

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNlbTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- MasterInstanceType

- RhcosAmi

- IgnitionLocation

- CertificateAuthorities

- MasterSecurityGroupId

- MasterInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- Master0Subnet

- Master1Subnet

- Master2Subnet

- Label:

default: "DNS"

Parameters:

- AutoRegisterDNS

- PrivateHostedZoneName

- PrivateHostedZoneId

- Label:

default: "Load Balancer Automation"

Parameters:

- AutoRegisterELB

- RegisterNlbTargetsLambdaArn

- ExternalApiTargetGroupArn

```

- InternalApiTargetGroupArn
- InternalServiceTargetGroupArn
ParameterLabels:
  InfrastructureName:
    default: "Infrastructure Name"
  VpcId:
    default: "VPC ID"
  Master0Subnet:
    default: "Master-0 Subnet"
  Master1Subnet:
    default: "Master-1 Subnet"
  Master2Subnet:
    default: "Master-2 Subnet"
  MasterInstanceType:
    default: "Master Instance Type"
  MasterInstanceProfileName:
    default: "Master Instance Profile Name"
  RhcosAmi:
    default: "Red Hat Enterprise Linux CoreOS AMI ID"
  BootstrapIgnitionLocation:
    default: "Master Ignition Source"
  CertificateAuthorities:
    default: "Ignition CA String"
  MasterSecurityGroupId:
    default: "Master Security Group ID"
  AutoRegisterDNS:
    default: "Use Provided DNS Automation"
  AutoRegisterELB:
    default: "Use Provided ELB Automation"
  PrivateHostedZoneName:
    default: "Private Hosted Zone Name"
  PrivateHostedZoneId:
    default: "Private Hosted Zone ID"

```

Conditions:

```

DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

```

Resources:

```

Master0:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"

```

```

UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{},"version":"2.2.0"},"networkd":{"},"passwd":{"},"storage":{"},"systemd":{"}}'
    - {
      SOURCE: !Ref IgnitionLocation,
      CA_BUNDLE: !Ref CertificateAuthorities,
    }
  Tags:
    - Key: !Join ["/", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "shared"

```

```

RegisterMaster0:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

RegisterMaster0InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

RegisterMaster0InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

Master1:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
          - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
  UserData:
    Fn::Base64: !Sub

```

```

- {"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}],"timeouts":
{},"version":"2.2.0"},"networkd":{"},"passwd":{"},"storage":{"},"systemd":{"}}
- {
  SOURCE: !Ref IgnitionLocation,
  CA_BUNDLE: !Ref CertificateAuthorities,
}
Tags:
- Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
  Value: "shared"

```

RegisterMaster1:

```

Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref ExternalApiTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

```

RegisterMaster1InternalApiTarget:

```

Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref InternalApiTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

```

RegisterMaster1InternalServiceTarget:

```

Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref InternalServiceTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

```

Master2:

```

Type: AWS::EC2::Instance
Properties:
  ImageId: !Ref RhcosAmi
  BlockDeviceMappings:
  - DeviceName: /dev/xvda
    Ebs:
      VolumeSize: "120"
      VolumeType: "gp2"
  IamInstanceProfile: !Ref MasterInstanceProfileName
  InstanceType: !Ref MasterInstanceType
  NetworkInterfaces:
  - AssociatePublicIp: "false"
    DeviceIndex: "0"
    GroupSet:
    - !Ref "MasterSecurityGroupId"
    SubnetId: !Ref "Master2Subnet"
  UserData:
  Fn::Base64: !Sub
    - {"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}],"timeouts":

```

```

{"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
- {
  SOURCE: !Ref IgnitionLocation,
  CA_BUNDLE: !Ref CertificateAuthorities,
}
Tags:
- Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
  Value: "shared"

RegisterMaster2:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

EtcDsrvRecords:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
    ResourceRecords:
      - !Join [
          " ",
          ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
        ]
      - !Join [
          " ",
          ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
        ]
      - !Join [
          " ",
          ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
        ]
    TTL: 60
    Type: SRV

EtcD0Record:

```



```

Condition: DoDns
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master0.PrivateIp
  TTL: 60
  Type: A

```

```

Etcd1Record:
Condition: DoDns
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master1.PrivateIp
  TTL: 60
  Type: A

```

```

Etcd2Record:
Condition: DoDns
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master2.PrivateIp
  TTL: 60
  Type: A

```

```

Outputs:
PrivateIPs:
  Description: The control-plane node private IP addresses.
  Value:
    !Join [
      ",",
      [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
    ]

```

2.9.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure

After you create all of the required infrastructure in Amazon Web Services (AWS), you can install the cluster.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- If you plan to manually manage the worker machines, create the worker machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

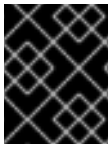
```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1  
--log-level=info 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

2.9.14.1. Creating the worker nodes in AWS

You can create worker nodes in Amazon Web Services (AWS) for your cluster to use. The easiest way to manually create these nodes is to modify the provided CloudFormation template.



IMPORTANT

The CloudFormation template creates a stack that represents one worker machine. You must create a stack for each worker machine.



NOTE

If you do not use the provided CloudFormation template to create your worker nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.

- Create the control plane machines.

Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
    10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.large" 16
  }
]
```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes.
- 4 Specify an **AWS::EC2::Image::Id** value.

- 5 A subnet, preferably private, to launch the worker nodes on.
- 6 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 7 The worker security group ID to associate with worker nodes.
- 8 Specify the **WorkerSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 9 The location to fetch bootstrap Ignition config file from.
- 10 Specify the generated Ignition config location, https://api-int.<cluster_name>.<domain_name>:22623/config/worker.
- 11 Base64 encoded certificate authority string to use.
- 12 Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 13 The IAM profile to associate with worker nodes.
- 14 Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 15 The type of AWS instance to use for the control plane machines.
- 16 Allowed values:
 - **m4.large**
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.large**
 - **c4.xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**
 - **c4.8xlarge**
 - **r4.large**
 - **r4.xlarge**
 - **r4.2xlarge**

- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**



IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

2. Copy the template from the **CloudFormation template for worker machines** section of this topic and save it as a YAML file on your computer. This template describes the networking objects and load balancers that your cluster requires.
3. If you specified an **m5** instance type as the value for **WorkerInstanceType**, add that instance type to the **WorkerInstanceType.AllowedValues** parameter in the CloudFormation template.
4. Create a worker stack.
 - a. Launch the template:



IMPORTANT

You must enter the command on a single line.

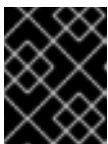
```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml \ 2
  --parameters file://<parameters>.json 3
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-workers**. You need the name of this stack if you remove the cluster.
- 2** **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3** **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- b. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

5. Continue to create worker stacks until you have created enough worker machines for your cluster.



IMPORTANT

You must create at least two worker machines, so you must create at least two stacks that use this CloudFormation template.

2.9.14.1.1. CloudFormation template for worker machines

You can use the following CloudFormation template to deploy the worker machines that you need for your OpenShift Container Platform cluster.

Example 2.26. CloudFormation template for worker machines

AWSTemplateFormatVersion: [2010-09-09](#)

Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:

InfrastructureName:

AllowedPattern: `^[a-zA-Z][a-zA-Z0-9\-\]{0,26}$`

MaxLength: `27`

MinLength: `1`

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of `27` characters.

Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.

Type: String

RhcosAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: `AWS::EC2::Image::Id`

Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: `AWS::EC2::Subnet::Id`

WorkerSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: `AWS::EC2::SecurityGroup::Id`

IgnitionLocation:

Default: `https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker`

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: `data:text/plain;charset=utf-8;base64,ABC...xYz==`

Description: Base64 encoded certificate authority string to use.

Type: String

WorkerInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

WorkerInstanceType:

Default: `m4.large`

Type: String

AllowedValues:

- `"m4.large"`
- `"m4.xlarge"`
- `"m4.2xlarge"`
- `"m4.4xlarge"`
- `"m4.8xlarge"`
- `"m4.10xlarge"`
- `"m4.16xlarge"`
- `"c4.large"`
- `"c4.xlarge"`
- `"c4.2xlarge"`
- `"c4.4xlarge"`
- `"c4.8xlarge"`
- `"r4.large"`

- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- WorkerInstanceType

- RhcosAmi

- IgnitionLocation

- CertificateAuthorities

- WorkerSecurityGroupId

- WorkerInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- Subnet

ParameterLabels:

Subnet:

default: "Subnet"

InfrastructureName:

default: "Infrastructure Name"

WorkerInstanceType:

default: "Worker Instance Type"

WorkerInstanceProfileName:

default: "Worker Instance Profile Name"

RhcosAmi:

default: "Red Hat Enterprise Linux CoreOS AMI ID"

IgnitionLocation:

default: "Worker Ignition Source"

CertificateAuthorities:

default: "Ignition CA String"

WorkerSecurityGroupId:

default: "Worker Security Group ID"

Resources:

Worker0:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

BlockDeviceMappings:

- DeviceName: /dev/xvda

Ebs:

VolumeSize: "120"

VolumeType: "gp2"

IamInstanceProfile: !Ref WorkerInstanceProfileName

InstanceType: !Ref WorkerInstanceType

```

NetworkInterfaces:
- AssociatePublicIpAddress: "false"
  DeviceIndex: "0"
  GroupSet:
- !Ref "WorkerSecurityGroupId"
  SubnetId: !Ref "Subnet"
UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}],"timeouts":{"version":"2.2.0"},"networkd":{"},"passwd":{"},"storage":{"},"systemd":{"}}}}}}'
    - {
      SOURCE: !Ref IgnitionLocation,
      CA_BUNDLE: !Ref CertificateAuthorities,
    }
  Tags:
- Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName] ]
  Value: "shared"

Outputs:
PrivateIP:
  Description: The compute node private IP address.
  Value: !GetAtt Worker0.PrivateIp

```

2.9.15. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

2.9.15.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:


```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.9.15.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.9.15.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.9.16. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

2.9.17. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
```

```

master-0 Ready    master 63m v1.18.3
master-1 Ready    master 63m v1.18.3
master-2 Ready    master 64m v1.18.3
worker-0 NotReady  worker 76s v1.18.3
worker-1 NotReady  worker 70s v1.18.3

```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME          AGE   REQUESTOR                                     CONDITION
csr-8b2br     15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps     15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

■

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}\n{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   73m   v1.20.0
master-1  Ready   master   73m   v1.20.0
master-2  Ready   master   74m   v1.20.0
worker-0  Ready   worker   11m   v1.20.0
worker-1  Ready   worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

2.9.18. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

2.9.18.1. Image registry storage configuration

Amazon Web Services provides default storage, which means the Image Registry Operator is available after installation. However, if the Registry Operator cannot create an S3 bucket and automatically configure storage, you must manually configure registry storage.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

You can configure registry storage for user-provisioned infrastructure in AWS to deploy OpenShift Container Platform to hidden regions. See [Configuring the registry for AWS user-provisioned infrastructure](#) for more information.

2.9.18.1.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an Amazon S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket and automatically configure storage, you can create an S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.
- For Amazon S3 storage, the secret is expected to contain two keys:
 - **REGISTRY_STORAGE_S3_ACCESSKEY**
 - **REGISTRY_STORAGE_S3_SECRETKEY**

Procedure

Use the following procedure if the Registry Operator cannot create an S3 bucket and automatically configure storage.

1. Set up a [Bucket Lifecycle Policy](#) to abort incomplete multipart uploads that are one day old.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

Example configuration

```
storage:  
  s3:  
    bucket: <bucket-name>  
    region: <region-name>
```



WARNING

To secure your registry images in AWS, [block public access](#) to the S3 bucket.

2.9.18.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

2.9.19. Deleting the bootstrap resources

After you complete the initial Operator configuration for the cluster, remove the bootstrap resources from Amazon Web Services (AWS).

Prerequisites

- You completed the initial Operator configuration for your cluster.

Procedure

1. Delete the bootstrap resources. If you used the CloudFormation template, [delete its stack](#):

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** is the name of your bootstrap stack.

2.9.20. Creating the Ingress DNS Records

If you removed the DNS Zone configuration, manually create DNS records that point to the Ingress load balancer. You can create either a wildcard record or specific records. While the following procedure uses A records, you can use other record types that you require, such as CNAME or alias.

Prerequisites

- You deployed an OpenShift Container Platform cluster on Amazon Web Services (AWS) that uses infrastructure that you provisioned.
- Install the OpenShift CLI (**oc**).
- Install the **jq** package.
- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#).

Procedure

1. Determine the routes to create.
 - To create a wildcard record, use ***.apps.<cluster_name>.<domain_name>**, where **<cluster_name>** is your cluster name, and **<domain_name>** is the Route 53 base domain for your OpenShift Container Platform cluster.
 - To create specific records, you must create a record for each route that your cluster uses, as shown in the output of the following command:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.<cluster_name>.<domain_name>
console-openshift-console.apps.<cluster_name>.<domain_name>
downloads-openshift-console.apps.<cluster_name>.<domain_name>
alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
```

2. Retrieve the Ingress Operator load balancer status and note the value of the external IP address that it uses, which is shown in the **EXTERNAL-IP** column:

```
$ oc -n openshift-ingress get service router-default
```

Example output

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
router-default	LoadBalancer	172.30.62.215	ab3...28.us-east-2.elb.amazonaws.com	80:31499/TCP,443:30693/TCP
		5m		

3. Locate the hosted zone ID for the load balancer:


```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName == "<external_ip>").CanonicalHostedZoneNameID' 1
```

- 1** For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer that you obtained.

Example output

```
Z3AADJGX6KTTL2
```

The output of this command is the load balancer hosted zone ID.

4. Obtain the public hosted zone ID for your cluster's domain:

```
$ aws route53 list-hosted-zones-by-name \
  --dns-name "<domain_name>" 1
  --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
  `<domain_name>.`].Id' 2
  --output text
```

- 1** **2** For **<domain_name>**, specify the Route 53 base domain for your OpenShift Container Platform cluster.

Example output

```
/hostedzone/Z3URY6TWQ91KVV
```

The public hosted zone ID for your domain is shown in the command output. In this example, it is **Z3URY6TWQ91KVV**.

5. Add the alias records to your private zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
> "Changes": [
> {
>   "Action": "CREATE",
>   "ResourceRecordSet": {
>     "Name": "\\052.apps.<cluster_domain>", 2
>     "Type": "A",
>     "AliasTarget":{
>       "HostedZoneId": "<hosted_zone_id>", 3
>       "DNSName": "<external_ip>.", 4
>       "EvaluateTargetHealth": false
>     }
>   }
> }
> ]
> }'
```

- 1 For **<private_hosted_zone_id>**, specify the value from the output of the CloudFormation template for DNS and load balancing.
- 2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

6. Add the records to your public zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>" --
change-batch '{ 1
> "Changes": [
> {
>   "Action": "CREATE",
>   "ResourceRecordSet": {
>     "Name": "\\052.apps.<cluster_domain>", 2
>     "Type": "A",
>     "AliasTarget":{
>       "HostedZoneId": "<hosted_zone_id>", 3
>       "DNSName": "<external_ip>.", 4
>       "EvaluateTargetHealth": false
>     }
>   }
> }
> ]
> }'
```

- 1 For **<public_hosted_zone_id>**, specify the public hosted zone for your domain.
- 2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

2.9.21. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, monitor the deployment to completion.

Prerequisites

- Removed the bootstrap node for an OpenShift Container Platform cluster on user-provisioned AWS infrastructure.

- Install the **oc** CLI and log in.

Procedure

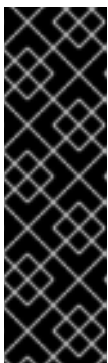
- Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```



IMPORTANT

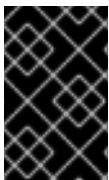
The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2.9.22. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- If necessary, you can [remove cloud provider credentials](#).

2.10. INSTALLING A CLUSTER ON AWS IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.5, you can install a cluster on Amazon Web Services (AWS) using infrastructure that you provide and an internal mirror of the installation release content.



IMPORTANT

While you can install an OpenShift Container Platform cluster by using mirrored installation release content, your cluster still requires Internet access to use the AWS APIs.

One way to create this infrastructure is to use the provided CloudFormation templates. You can modify the templates to customize your infrastructure or use the information that they contain to create AWS objects according to your company's policies.

**IMPORTANT**

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several CloudFormation templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

2.10.1. Prerequisites

- You [created a mirror registry on your mirror host](#) and obtained the **imageContentSources** data for your version of OpenShift Container Platform.

**IMPORTANT**

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an AWS account](#) to host the cluster.

**IMPORTANT**

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-lived credentials. To generate appropriate keys, see [Managing Access Keys for IAM Users](#) in the AWS documentation. You can supply the keys when you run the installation program.

- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#) in the AWS documentation.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.

**NOTE**

Be sure to also review this site list if you are configuring a proxy.

- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

2.10.2. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

2.10.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

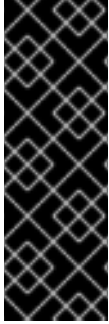
2.10.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

**IMPORTANT**

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

2.10.4. Required AWS infrastructure components

To install OpenShift Container Platform on user-provisioned infrastructure in Amazon Web Services (AWS), you must manually create both the machines and their supporting infrastructure.

For more information about the integration testing for different platforms, see the [OpenShift Container Platform 4.x Tested Integrations](#) page.

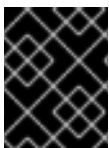
You can use the provided Cloud Formation templates to create this infrastructure, you can manually create the components, or you can reuse existing infrastructure that meets the cluster requirements. Review the Cloud Formation templates for more details about how the components interrelate.

2.10.4.1. Cluster machines

You need **AWS::EC2::Instance** objects for the following machines:

- A bootstrap machine. This machine is required during installation, but you can remove it after your cluster deploys.
- Three control plane machines. The control plane machines are not governed by a machine set.
- Compute machines. You must create at least two compute machines, which are also known as worker machines, during installation. These machines are not governed by a machine set.

You can use the following instance types for the cluster machines with the provided Cloud Formation templates.

**IMPORTANT**

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

Table 2.24. Instance types for machines

Instance type	Bootstrap	Control plane	Compute
i3.large	x		
m4.large or m5.large			x
m4.xlarge or m5.xlarge		x	x

Instance type	Bootstrap	Control plane	Compute
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.8xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
c4.large			x
c4.xlarge			x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x

You might be able to use other instance types that meet the specifications of these instance types.

2.10.4.2. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

2.10.4.3. Other infrastructure components

- A VPC
- DNS entries
- Load balancers (classic or network) and listeners
- A public and a private Route 53 zone
- Security groups
- IAM roles
- S3 buckets

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2 and ELB endpoints. To resolve this, you must create a VPC endpoint and attach it to the subnet that the clusters are using. The endpoints should be named as follows:

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

Component	AWS type	Description
VPC	<ul style="list-style-type: none"> • AWS::EC2::VPC • AWS::EC2::VPCEndpoint 	You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3.
Public subnets	<ul style="list-style-type: none"> • AWS::EC2::Subnet • AWS::EC2::SubnetNetworkACLAssociation 	Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules.

Component	AWS type	Description												
Internet gateway	<ul style="list-style-type: none"> ● AWS::EC2::InternetGateway ● AWS::EC2::VPCGatewayAttachment ● AWS::EC2::RouteTable ● AWS::EC2::Route ● AWS::EC2::SubnetRouteTableAssociation ● AWS::EC2::NatGateway ● AWS::EC2::EIP 	<p>You must have a public Internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the Internet and are not required for some restricted network or proxy scenarios.</p>												
Network access control	<ul style="list-style-type: none"> ● AWS::EC2::NetworkAcl ● AWS::EC2::NetworkAclEntry 	<p>You must allow the VPC to access the following ports:</p> <table border="1"> <thead> <tr> <th>Port</th> <th>Reason</th> </tr> </thead> <tbody> <tr> <td>80</td> <td>Inbound HTTP traffic</td> </tr> <tr> <td>443</td> <td>Inbound HTTPS traffic</td> </tr> <tr> <td>22</td> <td>Inbound SSH traffic</td> </tr> <tr> <td>1024 - 65535</td> <td>Inbound ephemeral traffic</td> </tr> <tr> <td>0 - 65535</td> <td>Outbound ephemeral traffic</td> </tr> </tbody> </table>	Port	Reason	80	Inbound HTTP traffic	443	Inbound HTTPS traffic	22	Inbound SSH traffic	1024 - 65535	Inbound ephemeral traffic	0 - 65535	Outbound ephemeral traffic
Port	Reason													
80	Inbound HTTP traffic													
443	Inbound HTTPS traffic													
22	Inbound SSH traffic													
1024 - 65535	Inbound ephemeral traffic													
0 - 65535	Outbound ephemeral traffic													
Private subnets	<ul style="list-style-type: none"> ● AWS::EC2::Subnet ● AWS::EC2::RouteTable ● AWS::EC2::SubnetRouteTableAssociation 	<p>Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them.</p>												

Required DNS and load balancing components

Your DNS and load balancer configuration needs to use a public hosted zone and can use a private hosted zone similar to the one that the installation program uses if it provisions the cluster's infrastructure. You must create a DNS entry that resolves to your load balancer. An entry for **api**.

<cluster_name>.<domain> must point to the external load balancer, and an entry for **api-int.<cluster_name>.<domain>** must point to the internal load balancer.

The cluster also requires load balancers and listeners for port 6443, which are required for the Kubernetes API and its extensions, and port 22623, which are required for the Ignition config files for new machines. The targets will be the master nodes. Port 6443 must be accessible to both clients external to the cluster and nodes within the cluster. Port 22623 must be accessible to nodes within the cluster.

Component	AWS type	Description
DNS	AWS::Route53::HostedZone	The hosted zone for your internal DNS.
etcd record sets	AWS::Route53::RecordSet	The registration records for etcd for your control plane machines.
Public load balancer	AWS::ElasticLoadBalancingV2::LoadBalancer	The load balancer for your public subnets.
External API server record	AWS::Route53::RecordSetGroup	Alias records for the external API server.
External listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 6443 for the external load balancer.
External target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the external load balancer.
Private load balancer	AWS::ElasticLoadBalancingV2::LoadBalancer	The load balancer for your private subnets.
Internal API server record	AWS::Route53::RecordSetGroup	Alias records for the internal API server.
Internal listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 22623 for the internal load balancer.

Component	AWS type	Description
Internal target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the internal load balancer.
Internal listener	AWS::ElasticLoadBalancingV2::Listener	A listener on port 6443 for the internal load balancer.
Internal target group	AWS::ElasticLoadBalancingV2::TargetGroup	The target group for the internal load balancer.

Security groups

The control plane and worker machines require access to the following ports:

Group	Type	IP Protocol	Port range
MasterSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
		tcp	6443
		tcp	22623
WorkerSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
BootstrapSecurityGroup	AWS::EC2::SecurityGroup	tcp	22
		tcp	19531

Control plane Ingress

The control plane machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

Ingress group	Description	IP protocol	Port range
MasterIngressEtcd	etcd	tcp	2379- 2380

Ingress group	Description	IP protocol	Port range
MasterIngress Vxlan	Vxlan packets	udp	4789
MasterIngress WorkerVxlan	Vxlan packets	udp	4789
MasterIngress Internal	Internal cluster communication and Kubernetes proxy metrics	tcp	9000 - 9999
MasterIngress WorkerInternal	Internal cluster communication	tcp	9000 - 9999
MasterIngress Kube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress WorkerKube	Kubernetes kubelet, scheduler and controller manager	tcp	10250 - 10259
MasterIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
MasterIngress WorkerIngressServices	Kubernetes Ingress services	tcp	30000 - 32767

Worker Ingress

The worker machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

Ingress group	Description	IP protocol	Port range
WorkerIngress Vxlan	Vxlan packets	udp	4789
WorkerIngress WorkerVxlan	Vxlan packets	udp	4789
WorkerIngress Internal	Internal cluster communication	tcp	9000 - 9999
WorkerIngress WorkerInternal	Internal cluster communication	tcp	9000 - 9999

Ingress group	Description	IP protocol	Port range
WorkerIngress Kube	Kubernetes kubelet, scheduler, and controller manager	tcp	10250
WorkerIngress WorkerKube	Kubernetes kubelet, scheduler, and controller manager	tcp	10250
WorkerIngress IngressServices	Kubernetes Ingress services	tcp	30000 - 32767
WorkerIngress WorkerIngress Services	Kubernetes Ingress services	tcp	30000 - 32767

Roles and instance profiles

You must grant the machines permissions in AWS. The provided CloudFormation templates grant the machines permission the following **AWS::IAM::Role** objects and provide a **AWS::IAM::InstanceProfile** for each set of roles. If you do not use the templates, you can grant the machines the following broad permissions or the following individual permissions.

Role	Effect	Action	Resource
Master	Allow	ec2:*	*
	Allow	elasticloadbalancing:*	*
	Allow	iam:PassRole	*
	Allow	s3:GetObject	*
Worker	Allow	ec2:Describe*	*
Bootstrap	Allow	ec2:Describe*	*
	Allow	ec2:AttachVolume	*
	Allow	ec2:DetachVolume	*

2.10.4.4. Required AWS permissions

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 2.27. Required EC2 permissions for installation

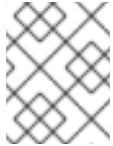
- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2>DeleteSecurityGroup**
- **ec2>DeleteSnapshot**
- **ec2>DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**

- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 2.28. Required permissions for creating network resources during installation

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**

- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



NOTE

If you use an existing VPC, your account does not require these permissions for creating network resources.

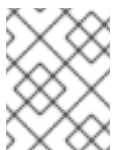
Example 2.29. Required Elastic Load Balancing permissions for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing>CreateListener**
- **elasticloadbalancing>CreateLoadBalancer**
- **elasticloadbalancing>CreateLoadBalancerListeners**
- **elasticloadbalancing>CreateTargetGroup**
- **elasticloadbalancing>DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**

- `elasticloadbalancing:ModifyTargetGroupAttributes`
- `elasticloadbalancing:RegisterInstancesWithLoadBalancer`
- `elasticloadbalancing:RegisterTargets`
- `elasticloadbalancing:SetLoadBalancerPoliciesOfListener`

Example 2.30. Required IAM permissions for installation

- `iam:AddRoleToInstanceProfile`
- `iam:CreateInstanceProfile`
- `iam:CreateRole`
- `iam:DeleteInstanceProfile`
- `iam>DeleteRole`
- `iam>DeleteRolePolicy`
- `iam:GetInstanceProfile`
- `iam:GetRole`
- `iam:GetRolePolicy`
- `iam:GetUser`
- `iam:ListInstanceProfilesForRole`
- `iam:ListRoles`
- `iam:ListUsers`
- `iam:PassRole`
- `iam:PutRolePolicy`
- `iam:RemoveRoleFromInstanceProfile`
- `iam:SimulatePrincipalPolicy`
- `iam:TagRole`



NOTE

If you have not created an elastic load balancer (ELB) in your AWS account, the IAM user also requires the `iam:CreateServiceLinkedRole` permission.

Example 2.31. Required Route 53 permissions for installation

- `route53:ChangeResourceRecordSets`

- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53>DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Example 2.32. Required S3 permissions for installation

- **s3:CreateBucket**
- **s3>DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**

- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

Example 2.33. S3 permissions that cluster Operators require

- **s3>DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

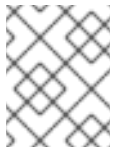
Example 2.34. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2>DeleteNetworkInterface**
- **ec2>DeleteVolume**
- **elasticloadbalancing>DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3>DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 2.35. Required permissions to delete network resources

- **ec2>DeleteDhcpOptions**

- **ec2:DeleteInternetGateway**
- **ec2>DeleteNatGateway**
- **ec2>DeleteRoute**
- **ec2>DeleteRouteTable**
- **ec2>DeleteSubnet**
- **ec2>DeleteVpc**
- **ec2>DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**

**NOTE**

If you use an existing VPC, your account does not require these permissions to delete network resources.

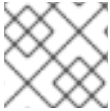
Example 2.36. Additional IAM and S3 permissions that are required to create manifests

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam>DeleteAccessKey**
- **iam>DeleteUser**
- **iam>DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam>ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam>ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**

- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

2.10.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

2.10.6. Creating the installation files for AWS

To install OpenShift Container Platform on Amazon Web Services (AWS) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files.

2.10.6.1. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.

Procedure

1. Obtain the **install-config.yaml** file.
 - a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **aws** as the platform to target.
 - iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.
 - iv. Select the AWS region to deploy the cluster to.
 - v. Select the base domain for the Route 53 service that you configured for your cluster.
 - vi. Enter a descriptive name for your cluster.
 - vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
```

3. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.
 - a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}'
```

For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name

and password for your mirror registry.

- b. Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  /-----END CERTIFICATE-----
```

- c. Add the image content resources:

```
imageContentSources:
- mirrors:
  - <local_registry>/<local_repository_name>/release
    source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

Use the **imageContentSources** section from the output of the command to mirror the repository or the values that you used when you mirrored the content from the media that you brought into your restricted network.

- d. Optional: Set the publishing strategy to **Internal**:

```
publish: Internal
```

By setting this option, you create an internal Ingress Controller and a private load balancer.

4. Optional: Back up the **install-config.yaml** file.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

2.10.6.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

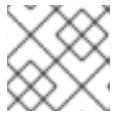
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

2.10.6.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
 - Locate the `mastersSchedulable` parameter and set its value to **False**.
 - Save and exit the file.

- Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}
```

- ❶ ❷ Remove this section completely.

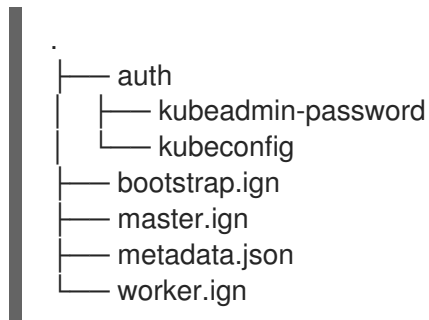
If you do so, you must add ingress DNS records manually in a later step.

- Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> ❶
```

- ❶ For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:



2.10.7. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Amazon Web Services (AWS). The provided CloudFormation templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

2.10.8. Creating a VPC in AWS

You must create a VPC in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements, including VPN and route tables. The easiest way to create the VPC is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.

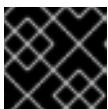
Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

- 1** The CIDR block for the VPC.
- 2** Specify a CIDR block in the format **x.x.x.x/16-24**.
- 3** The number of availability zones to deploy the VPC in.
- 4** Specify an integer between **1** and **3**.
- 5** The size of each subnet in each availability zone.
- 6** Specify an integer between **5** and **13**, where **5** is /27 and **13** is /19.

2. Copy the template from the **CloudFormation template for the VPC** section of this topic and save it as a YAML file on your computer. This template describes the VPC that your cluster requires.
3. Launch the template:



IMPORTANT

You must enter the command on a single line.

-

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.
- 2** **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3** **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

VpcId	The ID of your VPC.
PublicSubnetIds	The IDs of the new public subnets.
PrivateSubnetIds	The IDs of the new private subnets.

2.10.8.1. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

Example 2.37. CloudFormation template for the VPC

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(((0-9)[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.)\.{3}((0-9)[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
```

Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"

Type: Number

SubnetBits:

ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.

MinValue: 5

MaxValue: 13

Default: 12

Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"

Type: Number

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Network Configuration"

Parameters:

- VpcCidr

- SubnetBits

- Label:

default: "Availability Zones"

Parameters:

- AvailabilityZoneCount

ParameterLabels:

AvailabilityZoneCount:

default: "Availability Zone Count"

VpcCidr:

default: "VPC CIDR"

SubnetBits:

default: "Bits Per Subnet"

Conditions:

DoAz3: !Equals [3, !Ref AvailabilityZoneCount]

DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:

VPC:

Type: "AWS::EC2::VPC"

Properties:

EnableDnsSupport: "true"

EnableDnsHostnames: "true"

CidrBlock: !Ref VpcCidr

PublicSubnet:

Type: "AWS::EC2::Subnet"

Properties:

Vpclid: !Ref VPC

CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

AvailabilityZone: !Select

- 0

- Fn::GetAZs: !Ref "AWS::Region"

PublicSubnet2:

Type: "AWS::EC2::Subnet"

Condition: DoAz2

Properties:

Vpclid: !Ref VPC

CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]

```

    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PublicSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    Vpclid: !Ref VPC
    CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
GatewayToInternet:
  Type: "AWS::EC2::VPCGatewayAttachment"
  Properties:
    Vpclid: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
PublicRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    Vpclid: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    Vpclid: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 0
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:

```



```

Type: "AWS::EC2::RouteTable"
Properties:
  VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP
        - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:

```

```
AllocationId:
  "Fn::GetAtt":
    - EIP2
    - AllocationId
SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
    - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
        - EIP3
        - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:
  Type: "AWS::EC2::EIP"
  Condition: DoAz3
  Properties:
    Domain: vpc
Route3:
```

```

Type: "AWS::EC2::Route"
Condition: DoAz3
Properties:
  RouteTableId:
    Ref: PrivateRouteTable3
  DestinationCidrBlock: 0.0.0.0/0
  NatGatewayId:
    Ref: NAT3
S3Endpoint:
Type: AWS::EC2::VPCEndpoint
Properties:
  PolicyDocument:
    Version: 2012-10-17
    Statement:
      - Effect: Allow
        Principal: '*'
        Action:
          - '*'
        Resource:
          - '*'
    RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
    ServiceName: !Join
      - "
      - - com.amazonaws.
      - - !Ref 'AWS::Region'
      - - .s3
    Vpclid: !Ref VPC

Outputs:
Vpclid:
  Description: ID of the new VPC.
  Value: !Ref VPC
PublicSubnetIds:
  Description: Subnet IDs of the public subnets.
  Value:
    !Join [
      " ",
      [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
    ]
PrivateSubnetIds:
  Description: Subnet IDs of the private subnets.
  Value:
    !Join [
      " ",
      [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
    ]

```

2.10.9. Creating networking and load balancing components in AWS

You must configure networking and load balancing (classic or network) in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. The easiest way to create these components is to modify the provided CloudFormation template, which also creates a hosted zone and subnet tags.

You can run the template multiple times within a single VPC.



NOTE

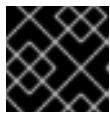
If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

Procedure

1. Obtain the Hosted Zone ID for the Route 53 zone that you specified in the **install-config.yaml** file for your cluster. You can obtain this ID from the AWS console or by running the following command:



IMPORTANT

You must enter the command on a single line.

```
$ aws route53 list-hosted-zones-by-name |
  jq --arg name "<route53_domain>." \ 1
  -r '.HostedZones | .[] | select(.Name=="\($name)") | .Id'
```

- 1 For the **<route53_domain>**, specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

2. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
    "ParameterValue": "<random_string>" 6
  }
]
```

```

},
{
  "ParameterKey": "HostedZoneName", 7
  "ParameterValue": "example.com" 8
},
{
  "ParameterKey": "PublicSubnets", 9
  "ParameterValue": "subnet-<random_string>" 10
},
{
  "ParameterKey": "PrivateSubnets", 11
  "ParameterValue": "subnet-<random_string>" 12
},
{
  "ParameterKey": "VpcId", 13
  "ParameterValue": "vpc-<random_string>" 14
}
]

```

- 1 A short, representative cluster name to use for host names, etc.
- 2 Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.
- 3 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 4 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 5 The Route 53 public zone ID to register the targets with.
- 6 Specify the Route 53 public zone ID, which as a format similar to **Z21IXYZABCZ2A4**. You can obtain this value from the AWS console.
- 7 The Route 53 zone to register the targets with.
- 8 Specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
- 9 The public subnets that you created for your VPC.
- 10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
- 11 The private subnets that you created for your VPC.
- 12 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
- 13 The VPC that you created for the cluster.
- 14 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

- Copy the template from the **CloudFormation template for the network and load balancers** section of this topic and save it as a YAML file on your computer. This template describes the networking and load balancing objects that your cluster requires.
- Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
  --capabilities CAPABILITY_NAMED_IAM
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-dns**. You need the name of this stack if you remove the cluster.
- 2** **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3** **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

PrivateHostedZoneId	Hosted zone ID for the private DNS.
ExternalApiLoadBalancerName	Full name of the external API load balancer.
InternalApiLoadBalancerName	Full name of the internal API load balancer.
ApiServerDnsName	Full host name of the API server.
RegisterNlbTargetsLambda	Lambda ARN useful to help register/deregister IP targets for these load balancers.

ExternalApiTargetGroupArn	ARN of external API target group.
InternalApiTargetGroupArn	ARN of internal API target group.
InternalServiceTargetGroupArn	ARN of internal service target group.

2.10.9.1. CloudFormation template for the network and load balancers

You can use the following CloudFormation template to deploy the networking objects and load balancers that you need for your OpenShift Container Platform cluster.

Example 2.38. CloudFormation template for the network and load balancers

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.

```

Type: List<AWS::EC2::Subnet::Id>
 PrivateSubnets:
 Description: The internal subnets.
 Type: List<AWS::EC2::Subnet::Id>
 VpcId:
 Description: The VPC-scoped resources will belong to this VPC.
 Type: AWS::EC2::VPC::Id

Metadata:

AWS::CloudFormation::Interface:
 ParameterGroups:
 - Label:
 default: "Cluster Information"
 Parameters:
 - ClusterName
 - InfrastructureName
 - Label:
 default: "Network Configuration"
 Parameters:
 - VpcId
 - PublicSubnets
 - PrivateSubnets
 - Label:
 default: "DNS"
 Parameters:
 - HostedZoneName
 - HostedZoneId
 ParameterLabels:
 ClusterName:
 default: "Cluster Name"
 InfrastructureName:
 default: "Infrastructure Name"
 VpcId:
 default: "VPC ID"
 PublicSubnets:
 default: "Public Subnets"
 PrivateSubnets:
 default: "Private Subnets"
 HostedZoneName:
 default: "Public Hosted Zone Name"
 HostedZoneId:
 default: "Public Hosted Zone ID"

Resources:

ExtApiElb:
 Type: AWS::ElasticLoadBalancingV2::LoadBalancer
 Properties:
 Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
 IpAddressType: ipv4
 Subnets: !Ref PublicSubnets
 Type: network

IntApiElb:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer
 Properties:
 Name: !Join ["-", [!Ref InfrastructureName, "int"]]


```

Scheme: internal
IpAddressType: ipv4
Subnets: !Ref PrivateSubnets
Type: network

```

IntDns:

```

Type: "AWS::Route53::HostedZone"
Properties:
  HostedZoneConfig:
    Comment: "Managed by CloudFormation"
    Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
  VPCs:
    - VPCId: !Ref Vpclid
      VPCRegion: !Ref "AWS::Region"

```

ExternalApiServerRecord:

```

Type: AWS::Route53::RecordSetGroup
Properties:
  Comment: Alias record for the API server
  HostedZoneId: !Ref HostedZoneId
  RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join [ "", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt ExtApiElb.DNSName

```

InternalApiServerRecord:

```

Type: AWS::Route53::RecordSetGroup
Properties:
  Comment: Alias record for the API server
  HostedZoneId: !Ref IntDns
  RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join [ "", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join [ "", [!Ref HostedZoneName, "."]]],
        ]

```

Type: A
AliasTarget:
 HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
 DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:

Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
 DefaultActions:
 - Type: forward
 TargetGroupArn:
 Ref: ExternalApiTargetGroup
 LoadBalancerArn:
 Ref: ExtApiElb
 Port: 6443
 Protocol: TCP

ExternalApiTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
 HealthCheckIntervalSeconds: 10
 HealthCheckPath: "/readyz"
 HealthCheckPort: 6443
 HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2
 Port: 6443
 Protocol: TCP
 TargetType: ip
 VpId:
 Ref: VpId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

InternalApiListener:

Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
 DefaultActions:
 - Type: forward
 TargetGroupArn:
 Ref: InternalApiTargetGroup
 LoadBalancerArn:
 Ref: IntApiElb
 Port: 6443
 Protocol: TCP

InternalApiTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup
Properties:
 HealthCheckIntervalSeconds: 10
 HealthCheckPath: "/readyz"
 HealthCheckPort: 6443
 HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2

Port: 6443
 Protocol: TCP
 TargetType: ip
 VpcId:
 Ref: VpcId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

InternalServiceInternalListener:
 Type: AWS::ElasticLoadBalancingV2::Listener
 Properties:
 DefaultActions:
 - Type: forward
 TargetGroupArn:
 Ref: InternalServiceTargetGroup
 LoadBalancerArn:
 Ref: IntApiElb
 Port: 22623
 Protocol: TCP

InternalServiceTargetGroup:
 Type: AWS::ElasticLoadBalancingV2::TargetGroup
 Properties:
 HealthCheckIntervalSeconds: 10
 HealthCheckPath: "/healthz"
 HealthCheckPort: 22623
 HealthCheckProtocol: HTTPS
 HealthyThresholdCount: 2
 UnhealthyThresholdCount: 2
 Port: 22623
 Protocol: TCP
 TargetType: ip
 VpcId:
 Ref: VpcId
 TargetGroupAttributes:
 - Key: deregistration_delay.timeout_seconds
 Value: 60

RegisterTargetLambdalaRole:
 Type: AWS::IAM::Role
 Properties:
 RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
 AssumeRolePolicyDocument:
 Version: "2012-10-17"
 Statement:
 - Effect: "Allow"
 Principal:
 Service:
 - "lambda.amazonaws.com"
 Action:
 - "sts:AssumeRole"
 Path: "/"
 Policies:
 - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
 PolicyDocument:

```

Version: "2012-10-17"
Statement:
- Effect: "Allow"
  Action:
  [
    "elasticloadbalancing:RegisterTargets",
    "elasticloadbalancing:DeregisterTargets",
  ]
  Resource: !Ref InternalApiTargetGroup
- Effect: "Allow"
  Action:
  [
    "elasticloadbalancing:RegisterTargets",
    "elasticloadbalancing:DeregisterTargets",
  ]
  Resource: !Ref InternalServiceTargetGroup
- Effect: "Allow"
  Action:
  [
    "elasticloadbalancing:RegisterTargets",
    "elasticloadbalancing:DeregisterTargets",
  ]
  Resource: !Ref ExternalApiTargetGroup

```

RegisterNlbIpTargets:

```

Type: "AWS::Lambda::Function"
Properties:
  Handler: "index.handler"
  Role:
    Fn::GetAtt:
    - "RegisterTargetLambdalamRole"
    - "Arn"
  Code:
    ZipFile: |
      import json
      import boto3
      import cfnresponse
      def handler(event, context):
        elb = boto3.client('elbv2')
        if event['RequestType'] == 'Delete':
          elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
[TargetArn],Targets=[{'Id': event['ResourceProperties']['TargetIp']})
          elif event['RequestType'] == 'Create':
            elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']})
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
  Runtime: "python3.7"
  Timeout: 120

```

RegisterSubnetTagsLambdalamRole:

```

Type: AWS::IAM::Role
Properties:
  RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
  AssumeRolePolicyDocument:

```

```

Version: "2012-10-17"
Statement:
- Effect: "Allow"
Principal:
  Service:
  - "lambda.amazonaws.com"
Action:
- "sts:AssumeRole"
Path: "/"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
PolicyDocument:
  Version: "2012-10-17"
  Statement:
  - Effect: "Allow"
    Action:
    [
      "ec2:DeleteTags",
      "ec2:CreateTags"
    ]
    Resource: "arn:aws:ec2:*:*:subnet/*"
  - Effect: "Allow"
    Action:
    [
      "ec2:DescribeSubnets",
      "ec2:DescribeTags"
    ]
    Resource: ""

```

RegisterSubnetTags:

```

Type: "AWS::Lambda::Function"
Properties:
  Handler: "index.handler"
  Role:
  Fn::GetAtt:
  - "RegisterSubnetTagsLambdalamRole"
  - "Arn"
Code:
  ZipFile: |
import json
import boto3
import cfnresponse
def handler(event, context):
  ec2_client = boto3.client('ec2')
  if event['RequestType'] == 'Delete':
    for subnet_id in event['ResourceProperties']['Subnets']:
      ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
    elif event['RequestType'] == 'Create':
      for subnet_id in event['ResourceProperties']['Subnets']:
        ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
        responseData = {}
        cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
  Runtime: "python3.7"

```

Timeout: 120

RegisterPublicSubnetTags:

Type: Custom::SubnetRegister

Properties:

ServiceToken: !GetAtt RegisterSubnetTags.Arn

InfrastructureName: !Ref InfrastructureName

Subnets: !Ref PublicSubnets

RegisterPrivateSubnetTags:

Type: Custom::SubnetRegister

Properties:

ServiceToken: !GetAtt RegisterSubnetTags.Arn

InfrastructureName: !Ref InfrastructureName

Subnets: !Ref PrivateSubnets

Outputs:

PrivateHostedZoneId:

Description: Hosted zone ID for the private DNS, which is required for private records.

Value: !Ref IntDns

ExternalApiLoadBalancerName:

Description: Full name of the external API load balancer.

Value: !GetAtt ExtApiElb.LoadBalancerFullName

InternalApiLoadBalancerName:

Description: Full name of the internal API load balancer.

Value: !GetAtt IntApiElb.LoadBalancerFullName

ApiServerDnsName:

Description: Full hostname of the API server, which is required for the Ignition config files.

Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]

RegisterNlbPTargetsLambda:

Description: Lambda ARN useful to help register or deregister IP targets for these load balancers.

Value: !GetAtt RegisterNlbPTargets.Arn

ExternalApiTargetGroupArn:

Description: ARN of the external API target group.

Value: !Ref ExternalApiTargetGroup

InternalApiTargetGroupArn:

Description: ARN of the internal API target group.

Value: !Ref InternalApiTargetGroup

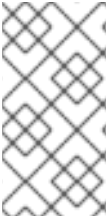
InternalServiceTargetGroupArn:

Description: ARN of the internal service target group.

Value: !Ref InternalServiceTargetGroup

2.10.10. Creating security group and roles in AWS

You must create security groups and roles in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. The easiest way to create these components is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.

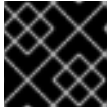
Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 The CIDR block for the VPC.
- 4 Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.
- 5 The private subnets that you created for your VPC.
- 6 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
- 7 The VPC that you created for the cluster.

- 8 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
2. Copy the template from the **CloudFormation template for security objects** section of this topic and save it as a YAML file on your computer. This template describes the security groups and roles that your cluster requires.
3. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
  --capabilities CAPABILITY_NAMED_IAM
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-sec**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

MasterSecurityGroupID	Master Security Group ID
WorkerSecurityGroupID	Worker Security Group ID
MasterInstanceProfile	Master IAM Instance Profile
WorkerInstanceProfile	Worker IAM Instance Profile

2.10.10.1. CloudFormation template for security objects

You can use the following CloudFormation template to deploy the security objects that you need for your OpenShift Container Platform cluster.

Example 2.39. CloudFormation template for security objects

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
    used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(((0-9)|1-9|0-9|10-9|2|20-4|0-9|250-5))\.\{3\}((0-9)|1-9|0-9|10-9|2|20-
    4|0-9|250-5))(V(1[6-9]|20-4))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: "Cluster Information"
        Parameters:
          - InfrastructureName
      - Label:
          default: "Network Configuration"
        Parameters:
          - VpcId
          - VpcCidr
          - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

```

Resources:

MasterSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Cluster Master Security Group

SecurityGroupIngress:

- IpProtocol: icmp

FromPort: 0

ToPort: 0

CidrIp: !Ref VpcCidr

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: !Ref VpcCidr

- IpProtocol: tcp

ToPort: 6443

FromPort: 6443

CidrIp: !Ref VpcCidr

- IpProtocol: tcp

FromPort: 22623

ToPort: 22623

CidrIp: !Ref VpcCidr

VpcId: !Ref VpcId

WorkerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Cluster Worker Security Group

SecurityGroupIngress:

- IpProtocol: icmp

FromPort: 0

ToPort: 0

CidrIp: !Ref VpcCidr

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: !Ref VpcCidr

VpcId: !Ref VpcId

MasterIngressEtcD:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: etcd

FromPort: 2379

ToPort: 2380

IpProtocol: tcp

MasterIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Vxlan packets

FromPort: 4789
ToPort: 4789
IpProtocol: udp

MasterIngressWorkerVxlan:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Vxlan packets
FromPort: 4789
ToPort: 4789
IpProtocol: udp

MasterIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

MasterIngressWorkerGeneve:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

MasterIngressInternal:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: tcp

MasterIngressWorkerInternal:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: tcp

MasterIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: udp

MasterIngressWorkerInternalUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: udp

MasterIngressKube:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Kubernetes kubelet, scheduler and controller manager
FromPort: 10250
ToPort: 10259
IpProtocol: tcp

MasterIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Kubernetes kubelet, scheduler and controller manager
FromPort: 10250
ToPort: 10259
IpProtocol: tcp

MasterIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Kubernetes ingress services
FromPort: 30000
ToPort: 32767
IpProtocol: tcp

MasterIngressWorkerIngressServices:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt MasterSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Kubernetes ingress services
FromPort: 30000
ToPort: 32767

IpProtocol: tcp

MasterIngressIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt MasterSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: udp

WorkerIngressVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Vxlan packets

FromPort: 4789

ToPort: 4789

IpProtocol: udp

WorkerIngressMasterVxlan:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Vxlan packets

FromPort: 4789

ToPort: 4789

IpProtocol: udp

WorkerIngressGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Geneve packets

FromPort: 6081

ToPort: 6081

IpProtocol: udp

WorkerIngressMasterGeneve:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Geneve packets
FromPort: 6081
ToPort: 6081
IpProtocol: udp

WorkerIngressInternal:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: tcp

WorkerIngressMasterInternal:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: tcp

WorkerIngressInternalUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: udp

WorkerIngressMasterInternalUDP:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
Description: Internal cluster communication
FromPort: 9000
ToPort: 9999
IpProtocol: udp

WorkerIngressKube:

Type: AWS::EC2::SecurityGroupIngress
Properties:
GroupId: !GetAtt WorkerSecurityGroup.GroupId
SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
Description: Kubernetes secure kubelet port
FromPort: 10250
ToPort: 10250
IpProtocol: tcp

WorkerIngressWorkerKube:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Internal Kubernetes communication

FromPort: 10250

ToPort: 10250

IpProtocol: tcp

WorkerIngressIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

WorkerIngressMasterIngressServices:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: tcp

WorkerIngressIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: udp

WorkerIngressMasterIngressServicesUDP:

Type: AWS::EC2::SecurityGroupIngress

Properties:

GroupId: !GetAtt WorkerSecurityGroup.GroupId

SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId

Description: Kubernetes ingress services

FromPort: 30000

ToPort: 32767

IpProtocol: udp

MasterIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

```
- Effect: "Allow"
Principal:
  Service:
    - "ec2.amazonaws.com"
Action:
  - "sts:AssumeRole"
Policies:
- PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
PolicyDocument:
  Version: "2012-10-17"
  Statement:
    - Effect: "Allow"
      Action:
        - "ec2:AttachVolume"
        - "ec2:AuthorizeSecurityGroupIngress"
        - "ec2:CreateSecurityGroup"
        - "ec2:CreateTags"
        - "ec2:CreateVolume"
        - "ec2>DeleteSecurityGroup"
        - "ec2>DeleteVolume"
        - "ec2:Describe*"
        - "ec2:DetachVolume"
        - "ec2:ModifyInstanceAttribute"
        - "ec2:ModifyVolume"
        - "ec2:RevokeSecurityGroupIngress"
        - "elasticloadbalancing:AddTags"
        - "elasticloadbalancing:AttachLoadBalancerToSubnets"
        - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
        - "elasticloadbalancing:CreateListener"
        - "elasticloadbalancing:CreateLoadBalancer"
        - "elasticloadbalancing:CreateLoadBalancerPolicy"
        - "elasticloadbalancing:CreateLoadBalancerListeners"
        - "elasticloadbalancing:CreateTargetGroup"
        - "elasticloadbalancing:ConfigureHealthCheck"
        - "elasticloadbalancing>DeleteListener"
        - "elasticloadbalancing>DeleteLoadBalancer"
        - "elasticloadbalancing>DeleteLoadBalancerListeners"
        - "elasticloadbalancing>DeleteTargetGroup"
        - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
        - "elasticloadbalancing:DeregisterTargets"
        - "elasticloadbalancing:Describe*"
        - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
        - "elasticloadbalancing:ModifyListener"
        - "elasticloadbalancing:ModifyLoadBalancerAttributes"
        - "elasticloadbalancing:ModifyTargetGroup"
        - "elasticloadbalancing:ModifyTargetGroupAttributes"
        - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
        - "elasticloadbalancing:RegisterTargets"
        - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
        - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
        - "kms:DescribeKey"
      Resource: ""
```

```
MasterInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
```


Roles:

- Ref: "MasterIamRole"

WorkerIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Policies:

- PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action:

- "ec2:DescribeInstances"

- "ec2:DescribeRegions"

Resource: "*"

WorkerInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Roles:

- Ref: "WorkerIamRole"

Outputs:

MasterSecurityGroupId:

Description: Master Security Group ID

Value: !GetAtt MasterSecurityGroup.GroupId

WorkerSecurityGroupId:

Description: Worker Security Group ID

Value: !GetAtt WorkerSecurityGroup.GroupId

MasterInstanceProfile:

Description: Master IAM Instance Profile

Value: !Ref MasterInstanceProfile

WorkerInstanceProfile:

Description: Worker IAM Instance Profile

Value: !Ref WorkerInstanceProfile

2.10.11. RHCOS AMIs for the AWS infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) AMI for your Amazon Web Services (AWS) zone for your OpenShift Container Platform nodes.

Table 2.25. RHCOS AMIs

AWS zone	AWS AMI
ap-northeast-1	ami-0530d04240177f118
ap-northeast-2	ami-09e4cd700276785d2
ap-south-1	ami-0754b15d212830477
ap-southeast-1	ami-03b46cc4b1518c5a8
ap-southeast-2	ami-0a5b99ab2234a4e6a
ca-central-1	ami-012bc4ee3b6c673bc
eu-central-1	ami-02e08df1201f1c2f8
eu-north-1	ami-0309c9d2fadcb2d5a
eu-west-1	ami-0bdd69d8e7cd18188
eu-west-2	ami-0e610e967a62dbdfa
eu-west-3	ami-0e817e26f638a71ac
me-south-1	ami-024117d7c87b7ff08
sa-east-1	ami-08e62f746b94950c1
us-east-1	ami-077ede5bed2e431ea
us-east-2	ami-0f4ecf819275850dd
us-west-1	ami-0c4990e435bc6c5fe
us-west-2	ami-000d6e92357ac605c

2.10.12. Creating the bootstrap node in AWS

You must create the bootstrap node in Amazon Web Services (AWS) to use during OpenShift Container Platform cluster initialization. The easiest way to create this node is to modify the provided CloudFormation template.



NOTE

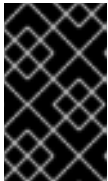
If you do not use the provided CloudFormation template to create your bootstrap node, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.

Procedure

1. Provide a location to serve the **bootstrap.ign** Ignition config file to your cluster. This file is located in your installation directory. One way to do this is to create an S3 bucket in your cluster's region and upload the Ignition config file to it.



IMPORTANT

The provided CloudFormation Template assumes that the Ignition config files for your cluster are served from an S3 bucket. If you choose to serve the files from another location, you must modify the templates.



NOTE

The bootstrap Ignition config file does contain secrets, like X.509 keys. The following steps provide basic security for the S3 bucket. To provide additional security, you can enable an S3 bucket policy to allow only certain users, such as the OpenShift IAM user, to access objects that the bucket contains. You can avoid S3 entirely and serve your bootstrap Ignition config file from any address that the bootstrap machine can reach.

- a. Create the bucket:

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

- 1 **<cluster-name>-infra** is the bucket name.

- b. Upload the **bootstrap.ign** Ignition config file to the bucket:

```
$ aws s3 cp bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign
```

- c. Verify that the file uploaded:

```
$ aws s3 ls s3://<cluster-name>-infra/
```

Example output

```
2019-04-03 16:15:16 314878 bootstrap.ign
```

2. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcoshAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroup", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:"
  }
]
```

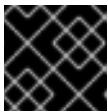
```

<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]

```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node.
- 4 Specify a valid **AWS::EC2::Image::Id** value.
- 5 CIDR block to allow SSH access to the bootstrap node.
- 6 Specify a CIDR block in the format **x.x.x.x/16-24**.
- 7 The public subnet that is associated with your VPC to launch the bootstrap node into.
- 8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.
- 9 The master security group ID (for registering temporary rules)
- 10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
- 11 The VPC created resources will belong to.
- 12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.
- 13 Location to fetch bootstrap Ignition config file from.
- 14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.
- 15 Whether or not to register a network load balancer (NLB).
- 16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
- 17 The ARN for NLB IP target registration lambda group.
- 18 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
- 19 The ARN for external API load balancer target group.
- 20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 21 The ARN for internal API load balancer target group.

- 22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
 - 23 The ARN for internal service load balancer target group.
 - 24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
3. Copy the template from the **CloudFormation template for the bootstrap machine** section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.
 4. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
  --capabilities CAPABILITY_NAMED_IAM
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-bootstrap**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

Bootstrap InstanceId	The bootstrap Instance ID.
Bootstrap PublicIp	The bootstrap node public IP address.
Bootstrap PrivateIp	The bootstrap node private IP address.

2.10.12.1. CloudFormation template for the bootstrap machine

You can use the following CloudFormation template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster.

Example 2.40. CloudFormation template for the bootstrap machine

```

AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
    maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
    used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
    4][0-9]|25[0-5])|([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
      - "yes"
      - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.

```

Type: String
InternalServiceTargetGroupArn:
Description: ARN for internal service load balancer target group.
Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:
default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:
default: "Host Information"

Parameters:

- RhcosAmi

- BootstrapIgnitionLocation

- MasterSecurityGroupId

- Label:
default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- PublicSubnet

- Label:
default: "Load Balancer Automation"

Parameters:

- AutoRegisterELB

- RegisterNlbTargetsLambdaArn

- ExternalApiTargetGroupArn

- InternalApiTargetGroupArn

- InternalServiceTargetGroupArn

ParameterLabels:

InfrastructureName:

default: "Infrastructure Name"

VpcId:

default: "VPC ID"

AllowedBootstrapSshCidr:

default: "Allowed SSH Source"

PublicSubnet:

default: "Public Subnet"

RhcosAmi:

default: "Red Hat Enterprise Linux CoreOS AMI ID"

BootstrapIgnitionLocation:

default: "Bootstrap Ignition Source"

MasterSecurityGroupId:

default: "Master Security Group ID"

AutoRegisterELB:

default: "Use Provided ELB Automation"

Conditions:

DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:

BootstrapIamRole:

Type: AWS::IAM::Role

Properties:

AssumeRolePolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Principal:

Service:

- "ec2.amazonaws.com"

Action:

- "sts:AssumeRole"

Path: "/"

Policies:

- PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]

PolicyDocument:

Version: "2012-10-17"

Statement:

- Effect: "Allow"

Action: "ec2:Describe*"

Resource: ""

- Effect: "Allow"

Action: "ec2:AttachVolume"

Resource: ""

- Effect: "Allow"

Action: "ec2:DetachVolume"

Resource: ""

- Effect: "Allow"

Action: "s3:GetObject"

Resource: ""

BootstrapInstanceProfile:

Type: "AWS::IAM::InstanceProfile"

Properties:

Path: "/"

Roles:

- Ref: "BootstrapIamRole"

BootstrapSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Cluster Bootstrap Security Group

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: !Ref AllowedBootstrapSshCidr

- IpProtocol: tcp

ToPort: 19531

FromPort: 19531

CidrIp: 0.0.0.0/0

Vpclid: !Ref Vpclid

BootstrapInstance:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

IamInstanceProfile: !Ref BootstrapInstanceProfile

```

InstanceType: "i3.large"
NetworkInterfaces:
- AssociatePublicIp: "true"
  DeviceIndex: "0"
  GroupSet:
  - !Ref "BootstrapSecurityGroup"
  - !Ref "MasterSecurityGroupId"
  SubnetId: !Ref "PublicSubnet"
UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"replace":{"source":"${S3Loc}","verification":{}}},"timeouts":
      {}, "version":"2.1.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
    - {
      S3Loc: !Ref BootstrapIgnitionLocation
    }

RegisterBootstrapApiTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref ExternalApiTargetGroupArn
  TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalApiTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref InternalApiTargetGroupArn
  TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalServiceTarget:
Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref InternalServiceTargetGroupArn
  TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
BootstrapInstanceid:
  Description: Bootstrap Instance ID.
  Value: !Ref BootstrapInstance

BootstrapPublicIp:
  Description: The bootstrap node public IP address.
  Value: !GetAtt BootstrapInstance.PublicIp

BootstrapPrivateIp:
  Description: The bootstrap node private IP address.
  Value: !GetAtt BootstrapInstance.PrivateIp

```

2.10.13. Creating the control plane machines in AWS

You must create the control plane machines in Amazon Web Services (AWS) for your cluster to use. The easiest way to create these nodes is to modify the provided CloudFormation template.



NOTE

If you do not use the provided CloudFormation template to create your control plane nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcOsAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
```

```

    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
  }, 20
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "m4.xlarge" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbPTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbPTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]

```

-
- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
 - 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
 - 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines.
 - 4 Specify an **AWS::EC2::Image::Id** value.
 - 5 Whether or not to perform DNS etcd registration.
 - 6 Specify **yes** or **no**. If you specify **yes**, you must provide hosted zone information.
 - 7 The Route 53 private zone ID to register the etcd targets with.
 - 8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.
 - 9 The Route 53 zone to register the targets with.
 - 10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route 53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.
 - 11 13 15 A subnet, preferably private, to launch the control plane machines on.
 - 12 14 16 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
 - 17 The master security group ID to associate with master nodes.
 - 18 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.
 - 19 The location to fetch control plane Ignition config file from.
 - 20 Specify the generated Ignition config file location, https://api-int.<cluster_name>.<domain_name>:22623/config/master.
 - 21 The base64 encoded certificate authority string to use.
 - 22 Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
 - 23 The IAM profile to associate with master nodes.
 - 24 Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
 - 25 The type of AWS instance to use for the control plane machines.
 - 26 Allowed values:
 - **m4.xlarge**

- **m4.2xlarge**
- **m4.4xlarge**
- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**

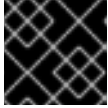


IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, specify an **m5** type, such as **m5.xlarge**, instead.

- 27 Whether or not to register a network load balancer (NLB).
- 28 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.
- 29 The ARN for NLB IP target registration lambda group.
- 30 Specify the **RegisterNlbTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing.
- 31 The ARN for external API load balancer target group.
- 32 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 33 The ARN for internal API load balancer target group.
- 34 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.
- 35 The ARN for internal service load balancer target group.
- 36 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing.

2. Copy the template from the **CloudFormation template for control plane machines** section of this topic and save it as a YAML file on your computer. This template describes the control plane machines that your cluster requires.
3. If you specified an **m5** instance type as the value for **MasterInstanceType**, add that instance type to the **MasterInstanceType.AllowedValues** parameter in the CloudFormation template.
4. Launch the template:



IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml 2
  --parameters file://<parameters>.json 3
```

- 1 **<name>** is the name for the CloudFormation stack, such as **cluster-control-plane**. You need the name of this stack if you remove the cluster.
- 2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2.10.13.1. CloudFormation template for control plane machines

You can use the following CloudFormation template to deploy the control plane machines that you need for your OpenShift Container Platform cluster.

Example 2.41. CloudFormation template for control plane machines

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-\_]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
```

Default: "yes"

AllowedValues:

- "yes"
- "no"

Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone information?

Type: String

PrivateHostedZoneId:

Description: The Route53 private zone ID to register the etcd targets with, such as Z21IXYZABCZ2A4.

Type: String

PrivateHostedZoneName:

Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit the trailing period.

Type: String

Master0Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master1Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

Master2Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: AWS::EC2::Subnet::Id

MasterSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: AWS::EC2::SecurityGroup::Id

IgnitionLocation:

Default: [https://api-int.\\$CLUSTER_NAME.\\$DOMAIN:22623/config/master](https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master)

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: data:text/plain;charset=utf-8;base64,ABC...xYz==

Description: Base64 encoded certificate authority string to use.

Type: String

MasterInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

MasterInstanceType:

Default: m4.xlarge

Type: String

AllowedValues:

- "m4.xlarge"
- "m4.2xlarge"
- "m4.4xlarge"
- "m4.8xlarge"
- "m4.10xlarge"
- "m4.16xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

AutoRegisterELB:

Default: "yes"

AllowedValues:

- "yes"
- "no"

Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?

Type: String

RegisterNlbTargetsLambdaArn:

Description: ARN for NLB IP target registration lambda. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

ExternalApiTargetGroupArn:

Description: ARN for external API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalApiTargetGroupArn:

Description: ARN for internal API load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

InternalServiceTargetGroupArn:

Description: ARN for internal service load balancer target group. Supply the value from the cluster infrastructure or select "no" for AutoRegisterELB.

Type: String

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- MasterInstanceType

- RhcosAmi

- IgnitionLocation

- CertificateAuthorities

- MasterSecurityGroupId

- MasterInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- VpcId

- AllowedBootstrapSshCidr

- Master0Subnet

- Master1Subnet

- Master2Subnet

- Label:

default: "DNS"

Parameters:

- AutoRegisterDNS

- PrivateHostedZoneName

- PrivateHostedZoneId

- Label:

default: "Load Balancer Automation"

Parameters:

- AutoRegisterELB
- RegisterNlbTargetsLambdaArn
- ExternalApiTargetGroupArn
- InternalApiTargetGroupArn
- InternalServiceTargetGroupArn

ParameterLabels:

InfrastructureName:
default: "Infrastructure Name"

VpcId:
default: "VPC ID"

Master0Subnet:
default: "Master-0 Subnet"

Master1Subnet:
default: "Master-1 Subnet"

Master2Subnet:
default: "Master-2 Subnet"

MasterInstanceType:
default: "Master Instance Type"

MasterInstanceProfileName:
default: "Master Instance Profile Name"

RhcosAmi:
default: "Red Hat Enterprise Linux CoreOS AMI ID"

BootstrapIgnitionLocation:
default: "Master Ignition Source"

CertificateAuthorities:
default: "Ignition CA String"

MasterSecurityGroupId:
default: "Master Security Group ID"

AutoRegisterDNS:
default: "Use Provided DNS Automation"

AutoRegisterELB:
default: "Use Provided ELB Automation"

PrivateHostedZoneName:
default: "Private Hosted Zone Name"

PrivateHostedZoneId:
default: "Private Hosted Zone ID"

Conditions:

DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:

Master0:

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

BlockDeviceMappings:

- DeviceName: /dev/xvda

Ebs:

VolumeSize: "120"

VolumeType: "gp2"

IamInstanceProfile: !Ref MasterInstanceProfileName

InstanceType: !Ref MasterInstanceType

NetworkInterfaces:

- AssociatePublicIp: "false"

```

DeviceIndex: "0"
GroupSet:
- !Ref "MasterSecurityGroupId"
SubnetId: !Ref "Master0Subnet"
UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":{},"version":"2.2.0"},"networkd":{"passwd":{},"storage":{},"systemd":{}}}'
    - {
      SOURCE: !Ref IgnitionLocation,
      CA_BUNDLE: !Ref CertificateAuthorities,
    }
  Tags:
  - Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
    Value: "shared"

```

```

RegisterMaster0:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

RegisterMaster0InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

RegisterMaster0InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master0.PrivateIp

```

```

Master1:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    BlockDeviceMappings:
    - DeviceName: /dev/xvda
      Ebs:
        VolumeSize: "120"
        VolumeType: "gp2"
    IamInstanceProfile: !Ref MasterInstanceProfileName
    InstanceType: !Ref MasterInstanceType
    NetworkInterfaces:
    - AssociatePublicIp: "false"
      DeviceIndex: "0"
      GroupSet:

```

```

- !Ref "MasterSecurityGroupId"
  SubnetId: !Ref "Master1Subnet"
  UserData:
    Fn::Base64: !Sub
      - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{"version":"2.2.0"},"networkd":{"passwd":{},"storage":{},"systemd":{}}
- {
  SOURCE: !Ref IgnitionLocation,
  CA_BUNDLE: !Ref CertificateAuthorities,
}
  Tags:
  - Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
    Value: "shared"

```

RegisterMaster1:

```

Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref ExternalApiTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

```

RegisterMaster1InternalApiTarget:

```

Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref InternalApiTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

```

RegisterMaster1InternalServiceTarget:

```

Condition: DoRegistration
Type: Custom::NLBRegister
Properties:
  ServiceToken: !Ref RegisterNlbTargetsLambdaArn
  TargetArn: !Ref InternalServiceTargetGroupArn
  TargetIp: !GetAtt Master1.PrivateIp

```

Master2:

```

Type: AWS::EC2::Instance
Properties:
  ImageId: !Ref RhcosAmi
  BlockDeviceMappings:
  - DeviceName: /dev/xvda
    Ebs:
      VolumeSize: "120"
      VolumeType: "gp2"
  IamInstanceProfile: !Ref MasterInstanceProfileName
  InstanceType: !Ref MasterInstanceType
  NetworkInterfaces:
  - AssociatePublicIpAddress: "false"
    DeviceIndex: "0"
    GroupSet:
    - !Ref "MasterSecurityGroupId"
  SubnetId: !Ref "Master2Subnet"

```

```

UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}],"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":
{},"version":"2.2.0"},"networkd":{"passwd":{},"storage":{},"systemd":{}}'
    - {
      SOURCE: !Ref IgnitionLocation,
      CA_BUNDLE: !Ref CertificateAuthorities,
    }
  Tags:
    - Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "shared"

```

```

RegisterMaster2:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

```

```

RegisterMaster2InternalApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

```

```

RegisterMaster2InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

```

```

EtcdSrvRecords:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join [ ".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
    ResourceRecords:
      - !Join [
          " ",
          ["0 10 2380", !Join [ ".", ["etcd-0", !Ref PrivateHostedZoneName]]],
        ]
      - !Join [
          " ",
          ["0 10 2380", !Join [ ".", ["etcd-1", !Ref PrivateHostedZoneName]]],
        ]
      - !Join [
          " ",
          ["0 10 2380", !Join [ ".", ["etcd-2", !Ref PrivateHostedZoneName]]],
        ]

```

```
TTL: 60
Type: SRV
```

Etcd0Record:

```
Condition: DoDns
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master0.PrivateIp
    TTL: 60
    Type: A
```

Etcd1Record:

```
Condition: DoDns
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master1.PrivateIp
    TTL: 60
    Type: A
```

Etcd2Record:

```
Condition: DoDns
Type: AWS::Route53::RecordSet
Properties:
  HostedZoneId: !Ref PrivateHostedZoneId
  Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
  ResourceRecords:
    - !GetAtt Master2.PrivateIp
    TTL: 60
    Type: A
```

Outputs:

PrivateIPs:

```
Description: The control-plane node private IP addresses.
Value:
  !Join [
    ",",
    [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
  ]
```

2.10.14. Initializing the bootstrap node on AWS with user-provisioned infrastructure

After you create all of the required infrastructure in Amazon Web Services (AWS), you can install the cluster.

Prerequisites

- Configure an AWS account.

- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- If you plan to manually manage the worker machines, create the worker machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

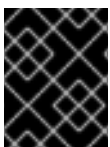
```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

2.10.14.1. Creating the worker nodes in AWS

You can create worker nodes in Amazon Web Services (AWS) for your cluster to use. The easiest way to manually create these nodes is to modify the provided CloudFormation template.



IMPORTANT

The CloudFormation template creates a stack that represents one worker machine. You must create a stack for each worker machine.



NOTE

If you do not use the provided CloudFormation template to create your worker nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an AWS account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in AWS.
- Create and configure DNS, load balancers, and listeners in AWS.

- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcocAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupID", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
  } 10
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.large" 16
  }
]
```

- 1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
- 2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

- 3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes.
- 4 Specify an **AWS::EC2::Image::Id** value.
- 5 A subnet, preferably private, to launch the worker nodes on.
- 6 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.
- 7 The worker security group ID to associate with worker nodes.
- 8 Specify the **WorkerSecurityGroupID** value from the output of the CloudFormation template for the security group and roles.
- 9 The location to fetch bootstrap Ignition config file from.
- 10 Specify the generated Ignition config location, https://api-int.<cluster_name>.<domain_name>:22623/config/worker.
- 11 Base64 encoded certificate authority string to use.
- 12 Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC...xYz==**.
- 13 The IAM profile to associate with worker nodes.
- 14 Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.
- 15 The type of AWS instance to use for the control plane machines.
- 16 Allowed values:
 - **m4.large**
 - **m4.xlarge**
 - **m4.2xlarge**
 - **m4.4xlarge**
 - **m4.8xlarge**
 - **m4.10xlarge**
 - **m4.16xlarge**
 - **c4.large**
 - **c4.xlarge**
 - **c4.2xlarge**
 - **c4.4xlarge**
 - **c4.8xlarge**
 - **r4.large**

- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**



IMPORTANT

If **m4** instance types are not available in your region, such as with **eu-west-3**, use **m5** types instead.

2. Copy the template from the **CloudFormation template for worker machines** section of this topic and save it as a YAML file on your computer. This template describes the networking objects and load balancers that your cluster requires.
3. If you specified an **m5** instance type as the value for **WorkerInstanceType**, add that instance type to the **WorkerInstanceType.AllowedValues** parameter in the CloudFormation template.
4. Create a worker stack.
 - a. Launch the template:



IMPORTANT

You must enter the command on a single line.

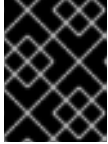
```
$ aws cloudformation create-stack --stack-name <name> 1
  --template-body file://<template>.yaml \ 2
  --parameters file://<parameters>.json 3
```

- 1** **<name>** is the name for the CloudFormation stack, such as **cluster-workers**. You need the name of this stack if you remove the cluster.
- 2** **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
- 3** **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

- b. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

5. Continue to create worker stacks until you have created enough worker machines for your cluster.

**IMPORTANT**

You must create at least two worker machines, so you must create at least two stacks that use this CloudFormation template.

2.10.14.1.1. CloudFormation template for worker machines

You can use the following CloudFormation template to deploy the worker machines that you need for your OpenShift Container Platform cluster.

Example 2.42. CloudFormation template for worker machines

AWSTemplateFormatVersion: **2010-09-09**

Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:

InfrastructureName:

AllowedPattern: `^[a-zA-Z][a-zA-Z0-9-]{0,26}$`

MaxLength: **27**

MinLength: **1**

ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of **27** characters.

Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.

Type: String

RhcosAmi:

Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.

Type: `AWS::EC2::Image::Id`

Subnet:

Description: The subnets, recommend private, to launch the master nodes into.

Type: `AWS::EC2::Subnet::Id`

WorkerSecurityGroupId:

Description: The master security group ID to associate with master nodes.

Type: `AWS::EC2::SecurityGroup::Id`

IgnitionLocation:

Default: `https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker`

Description: Ignition config file location.

Type: String

CertificateAuthorities:

Default: `data:text/plain;charset=utf-8;base64,ABC...xYz==`

Description: Base64 encoded certificate authority string to use.

Type: String

WorkerInstanceProfileName:

Description: IAM profile to associate with master nodes.

Type: String

WorkerInstanceType:

Default: `m4.large`

Type: String

AllowedValues:

- `"m4.large"`

- `"m4.xlarge"`

- `"m4.2xlarge"`

- `"m4.4xlarge"`

- `"m4.8xlarge"`

- `"m4.10xlarge"`

- `"m4.16xlarge"`

- `"c4.large"`

- "c4.xlarge"
- "c4.2xlarge"
- "c4.4xlarge"
- "c4.8xlarge"
- "r4.large"
- "r4.xlarge"
- "r4.2xlarge"
- "r4.4xlarge"
- "r4.8xlarge"
- "r4.16xlarge"

Metadata:**AWS::CloudFormation::Interface:****ParameterGroups:****- Label:**

default: "Cluster Information"

Parameters:

- InfrastructureName

- Label:

default: "Host Information"

Parameters:

- WorkerInstanceType
- RhcosAmi
- IgnitionLocation
- CertificateAuthorities
- WorkerSecurityGroupId
- WorkerInstanceProfileName

- Label:

default: "Network Configuration"

Parameters:

- Subnet

ParameterLabels:**Subnet:**

default: "Subnet"

InfrastructureName:

default: "Infrastructure Name"

WorkerInstanceType:

default: "Worker Instance Type"

WorkerInstanceProfileName:

default: "Worker Instance Profile Name"

RhcosAmi:

default: "Red Hat Enterprise Linux CoreOS AMI ID"

IgnitionLocation:

default: "Worker Ignition Source"

CertificateAuthorities:

default: "Ignition CA String"

WorkerSecurityGroupId:

default: "Worker Security Group ID"

Resources:**Worker0:**

Type: AWS::EC2::Instance

Properties:

ImageId: !Ref RhcosAmi

BlockDeviceMappings:

- DeviceName: /dev/xvda

```

Ebs:
  VolumeSize: "120"
  VolumeType: "gp2"
IamInstanceProfile: !Ref WorkerInstanceProfileName
InstanceType: !Ref WorkerInstanceType
NetworkInterfaces:
- AssociatePublicIpAddress: "false"
  DeviceIndex: "0"
  GroupSet:
  - !Ref "WorkerSecurityGroupId"
  SubnetId: !Ref "Subnet"
UserData:
  Fn::Base64: !Sub
    - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]},"timeouts":{"version":"2.2.0"},"networkd":{"passwd":{"storage":{"systemd":{"SOURCE: !Ref IgnitionLocation,
      CA_BUNDLE: !Ref CertificateAuthorities,
    }
  }
Tags:
- Key: !Join [ "", ["kubernetes.io/cluster/", !Ref InfrastructureName]
  Value: "shared"

Outputs:
PrivatelP:
  Description: The compute node private IP address.
  Value: !GetAtt Worker0.PrivateIp

```

2.10.15. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:
 -

```
$ oc whoami
```

Example output

```
system:admin
```

2.10.16. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 63m  v1.18.3
master-1  Ready   master 63m  v1.18.3
master-2  Ready   master 64m  v1.18.3
worker-0  NotReady worker 76s  v1.18.3
worker-1  NotReady worker 70s  v1.18.3
```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE  REQUESTOR                                     CONDITION
csr-8b2br  15m  system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
csr-8vnps  15m  system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   73m   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

2.10.17. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s

network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

2.10.17.1. Image registry storage configuration

Amazon Web Services provides default storage, which means the Image Registry Operator is available after installation. However, if the Registry Operator cannot create an S3 bucket and automatically configure storage, you must manually configure registry storage.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

2.10.17.1.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an Amazon S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket and automatically configure storage, you can create an S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.
- For Amazon S3 storage, the secret is expected to contain two keys:
 - **REGISTRY_STORAGE_S3_ACCESSKEY**
 - **REGISTRY_STORAGE_S3_SECRETKEY**

Procedure

Use the following procedure if the Registry Operator cannot create an S3 bucket and automatically configure storage.

1. Set up a [Bucket Lifecycle Policy](#) to abort incomplete multipart uploads that are one day old.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

Example configuration

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```



WARNING

To secure your registry images in AWS, [block public access](#) to the S3 bucket.

2.10.17.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.
 - Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

2.10.18. Deleting the bootstrap resources

After you complete the initial Operator configuration for the cluster, remove the bootstrap resources from Amazon Web Services (AWS).

Prerequisites

- You completed the initial Operator configuration for your cluster.

Procedure

- Delete the bootstrap resources. If you used the CloudFormation template, [delete its stack](#):

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** is the name of your bootstrap stack.

2.10.19. Creating the Ingress DNS Records

If you removed the DNS Zone configuration, manually create DNS records that point to the Ingress load balancer. You can create either a wildcard record or specific records. While the following procedure uses A records, you can use other record types that you require, such as CNAME or alias.

Prerequisites

- You deployed an OpenShift Container Platform cluster on Amazon Web Services (AWS) that uses infrastructure that you provisioned.
- Install the OpenShift CLI (**oc**).
- Install the **jq** package.
- Download the AWS CLI and install it on your computer. See [Install the AWS CLI Using the Bundled Installer \(Linux, macOS, or Unix\)](#).

Procedure

- Determine the routes to create.
 - To create a wildcard record, use ***.apps.<cluster_name>.<domain_name>**, where **<cluster_name>** is your cluster name, and **<domain_name>** is the Route 53 base domain for your OpenShift Container Platform cluster.
 - To create specific records, you must create a record for each route that your cluster uses, as shown in the output of the following command:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}}{end}{end}' routes
```

Example output

```

oauth-openshift.apps.<cluster_name>.<domain_name>
console-openshift-console.apps.<cluster_name>.<domain_name>
downloads-openshift-console.apps.<cluster_name>.<domain_name>
alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>

```

- Retrieve the Ingress Operator load balancer status and note the value of the external IP address that it uses, which is shown in the **EXTERNAL-IP** column:

```
$ oc -n openshift-ingress get service router-default
```

Example output

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
router-default	LoadBalancer	172.30.62.215	ab3...28.us-east-2.elb.amazonaws.com	80:31499/TCP,443:30693/TCP
		5m		

- Locate the hosted zone ID for the load balancer:

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName == "<external_ip>").CanonicalHostedZoneNameID' 1
```

- For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer that you obtained.

Example output

```
Z3AADJGX6KTTL2
```

The output of this command is the load balancer hosted zone ID.

- Obtain the public hosted zone ID for your cluster's domain:

```
$ aws route53 list-hosted-zones-by-name \
  --dns-name "<domain_name>" 1 \
  --query 'HostedZones[? Config.PrivateZone != `true` && Name == \
  `<domain_name>.`].Id' 2 \
  --output text
```

- For **<domain_name>**, specify the Route 53 base domain for your OpenShift Container Platform cluster.

Example output

```
/hostedzone/Z3URY6TWQ91KVV
```

The public hosted zone ID for your domain is shown in the command output. In this example, it is **Z3URY6TWQ91KVV**.

5. Add the alias records to your private zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
> "Changes": [
> {
>   "Action": "CREATE",
>   "ResourceRecordSet": {
>     "Name": "\\052.apps.<cluster_domain>", 2
>     "Type": "A",
>     "AliasTarget":{
>       "HostedZoneId": "<hosted_zone_id>", 3
>       "DNSName": "<external_ip>.", 4
>       "EvaluateTargetHealth": false
>     }
>   }
> }
> ]
> }'
```

- 1 For **<private_hosted_zone_id>**, specify the value from the output of the CloudFormation template for DNS and load balancing.
- 2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

6. Add the records to your public zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ 1
> "Changes": [
> {
>   "Action": "CREATE",
>   "ResourceRecordSet": {
>     "Name": "\\052.apps.<cluster_domain>", 2
>     "Type": "A",
>     "AliasTarget":{
>       "HostedZoneId": "<hosted_zone_id>", 3
>       "DNSName": "<external_ip>.", 4
>       "EvaluateTargetHealth": false
>     }
>   }
> }
> ]
> }'
```

- 1 For **<public_hosted_zone_id>**, specify the public hosted zone for your domain.
- 2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
- 3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
- 4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (.) in this parameter value.

2.10.20. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, monitor the deployment to completion.

Prerequisites

- Removed the bootstrap node for an OpenShift Container Platform cluster on user-provisioned AWS infrastructure.
- Install the **oc** CLI and log in.

Procedure

1. Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrap** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Register your cluster on the [Cluster registration](#) page.

2.10.21. Next steps

- [Customize your cluster.](#)

- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to [use Operator Lifecycle Manager \(OLM\) on restricted networks](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).
- If necessary, you can [remove cloud provider credentials](#).

2.11. UNINSTALLING A CLUSTER ON AWS

You can remove a cluster that you deployed to Amazon Web Services (AWS).

2.11.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access.

Prerequisites

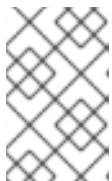
- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

CHAPTER 3. INSTALLING ON AZURE

3.1. CONFIGURING AN AZURE ACCOUNT

Before you can install OpenShift Container Platform, you must configure a Microsoft Azure account.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

3.1.1. Azure account limits

The OpenShift Container Platform cluster uses a number of Microsoft Azure components, and the default [Azure subscription and service limits, quotas, and constraints](#) affect your ability to install OpenShift Container Platform clusters.



IMPORTANT

Default limits vary by offer category types, such as Free Trial and Pay-As-You-Go, and by series, such as Dv2, F, and G. For example, the default for Enterprise Agreement subscriptions is 350 cores.

Check the limits for your subscription type and if necessary, increase quota limits for your account before you install a default cluster on Azure.

The following table summarizes the Azure components whose limits can impact your ability to install and run OpenShift Container Platform clusters.

Component	Number of components required by default	Default Azure limit	Description

Component	Number of components required by default	Default Azure limit	Description
vCPU	40	20 per region	<p>A default cluster requires 40 vCPUs, so you must increase the account limit.</p> <p>By default, each cluster creates the following instances:</p> <ul style="list-style-type: none"> • One bootstrap machine, which is removed after installation • Three control plane machines • Three compute machines <p>Because the bootstrap machine uses Standard_D4s_v3 machines, which use 4 vCPUs, the control plane machines use Standard_D8s_v3 virtual machines, which use 8 vCPUs, and the worker machines use Standard_D4s_v3 virtual machines, which use 4 vCPUs, a default cluster requires 40 vCPUs. The bootstrap node VM, which uses 4 vCPUs, is used only during installation.</p> <p>To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, you must further increase the vCPU limit for your account to ensure that your cluster can deploy the machines that you require.</p> <p>By default, the installation program distributes control plane and compute machines across all availability zones within a region. To ensure high availability for your cluster, select a region with at least three availability zones. If your region contains fewer than three availability zones, the installation program places more than one control plane machine in the available zones.</p>
VNet	1	1000 per region	Each default cluster requires one Virtual Network (VNet), which contains two subnets.
Network interfaces	6	65,536 per region	Each default cluster requires six network interfaces. If you create more machines or your deployed workloads create load balancers, your cluster uses more network interfaces.

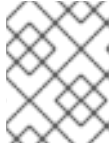
Component	Number of components required by default	Default Azure limit	Description						
Network security groups	2	5000	<p>Each default cluster creates network security groups for each subnet in the VNet. The default cluster creates network security groups for the control plane and for the compute node subnets:</p> <table border="1"> <tr> <td>control plane</td> <td>Allows the control plane machines to be reached on port 6443 from anywhere</td> </tr> <tr> <td>node</td> <td>Allows worker nodes to be reached from the Internet on ports 80 and 443</td> </tr> </table>	control plane	Allows the control plane machines to be reached on port 6443 from anywhere	node	Allows worker nodes to be reached from the Internet on ports 80 and 443		
control plane	Allows the control plane machines to be reached on port 6443 from anywhere								
node	Allows worker nodes to be reached from the Internet on ports 80 and 443								
Network load balancers	3	1000 per region	<p>Each cluster creates the following load balancers:</p> <table border="1"> <tr> <td>default</td> <td>Public IP address that load balances requests to ports 80 and 443 across worker machines</td> </tr> <tr> <td>internal</td> <td>Private IP address that load balances requests to ports 6443 and 22623 across control plane machines</td> </tr> <tr> <td>external</td> <td>Public IP address that load balances requests to port 6443 across control plane machines</td> </tr> </table> <p>If your applications create more Kubernetes LoadBalancer service objects, your cluster uses more load balancers.</p>	default	Public IP address that load balances requests to ports 80 and 443 across worker machines	internal	Private IP address that load balances requests to ports 6443 and 22623 across control plane machines	external	Public IP address that load balances requests to port 6443 across control plane machines
default	Public IP address that load balances requests to ports 80 and 443 across worker machines								
internal	Private IP address that load balances requests to ports 6443 and 22623 across control plane machines								
external	Public IP address that load balances requests to port 6443 across control plane machines								
Public IP addresses	3		Each of the two public load balancers uses a public IP address. The bootstrap machine also uses a public IP address so that you can SSH into the machine to troubleshoot issues during installation. The IP address for the bootstrap node is used only during installation.						
Private IP addresses	7		The internal load balancer, each of the three control plane machines, and each of the three worker machines each use a private IP address.						

3.1.2. Configuring a public DNS zone in Azure

To install OpenShift Container Platform, the Microsoft Azure account you use must have a dedicated public hosted DNS zone in your account. This zone must be authoritative for the domain. This service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through Azure or another source.



NOTE

For more information about purchasing domains through Azure, see [Buy a custom domain name for Azure App Service](#) in the Azure documentation.

2. If you are using an existing domain and registrar, migrate its DNS to Azure. See [Migrate an active DNS name to Azure App Service](#) in the Azure documentation.
3. Configure DNS for your domain. Follow the steps in the [Tutorial: Host your domain in Azure DNS](#) in the Azure documentation to create a public hosted zone for your domain or subdomain, extract the new authoritative name servers, and update the registrar records for the name servers that your domain uses.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
4. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

3.1.3. Increasing Azure account limits

To increase an account limit, file a support request on the Azure portal.



NOTE

You can increase only one type of quota per support request.

Procedure

1. From the Azure portal, click **Help + support** in the lower left corner.
2. Click **New support request** and then select the required values:
 - a. From the **Issue type** list, select **Service and subscription limits (quotas)**
 - b. From the **Subscription** list, select the subscription to modify.
 - c. From the **Quota type** list, select the quota to increase. For example, select **Compute-VM (cores-vCPUs) subscription limit increases** to increase the number of vCPUs, which is required to install a cluster.
 - d. Click **Next: Solutions**.
3. On the **Problem Details** page, provide the required information for your quota increase:
 - a. Click **Provide details** and provide the required details in the **Quota details** window.

- b. In the SUPPORT METHOD and CONTACT INFO sections, provide the issue severity and your contact details.
4. Click **Next: Review + create** and then click **Create**.

3.1.4. Required Azure roles

Your Microsoft Azure account must have the following roles for the subscription that you use:

- **User Access Administrator**

To set roles on the Azure portal, see the [Manage access to Azure resources using RBAC and the Azure portal](#) in the Azure documentation.

3.1.5. Creating a service principal

Because OpenShift Container Platform and its installation program must create Microsoft Azure resources through Azure Resource Manager, you must create a service principal to represent it.

Prerequisites

- Install or update the [Azure CLI](#).
- Install the **jq** package.
- Your Azure account has the required roles for the subscription that you use.

Procedure

1. Log in to the Azure CLI:

```
$ az login
```

Log in to Azure in the web console by using your credentials.

2. If your Azure account uses subscriptions, ensure that you are using the right subscription.
 - a. View the list of available accounts and record the **tenantId** value for the subscription you want to use for your cluster:

```
$ az account list --refresh
```

Example output

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

```
}
}
]
```

- b. View your active account details and confirm that the **tenantId** value matches the subscription you want to use:

```
$ az account show
```

Example output

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", 1
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

- 1 Ensure that the value of the **tenantId** parameter is the UUID of the correct subscription.

- c. If you are not using the right subscription, change the active subscription:

```
$ az account set -s <id> 1
```

- 1 Substitute the value of the **id** for the subscription that you want to use for **<id>**.

- d. If you changed the active subscription, display your account information again:

```
$ az account show
```

Example output

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. Record the values of the **tenantId** and **id** parameters from the previous output. You need these values during OpenShift Container Platform installation.
4. Create the service principal for your account:

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
```

- 1 Replace **<service_principal>** with the name to assign to the service principal.

Example output

```
Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the
required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
Retrying role assignment creation: 4/36
{
  "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
  "displayName": "<service_principal>",
  "name": "http://<service_principal>",
  "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
}
```

5. Record the values of the **appId** and **password** parameters from the previous output. You need these values during OpenShift Container Platform installation.
6. Grant additional permissions to the service principal. The service principal requires the legacy **Azure Active Directory Graph** → **Application.ReadWrite.OwnedBy** permission and the **User Access Administrator** role for the cluster to assign credentials for its components.
 - a. To assign the **User Access Administrator** role, run the following command:

```
$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \
  | jq '[0].objectId' -r) 1
```

- 1 Replace **<appId>** with the **appId** parameter value for your service principal.

- b. To assign the **Azure Active Directory Graph** permission, run the following command:

```
$ az ad app permission add --id <appId> \ 1
  --api 00000002-0000-0000-c000-000000000000 \
  --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
```

- 1 Replace **<appId>** with the **appId** parameter value for your service principal.

Example output

```
Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api
00000002-0000-0000-c000-000000000000" is needed to make the change effective
```

■

For more information about the specific permissions that you grant with this command, see the [GUID Table for Windows Azure Active Directory Permissions](#) .

- c. Approve the permissions request. If your account does not have the Azure Active Directory tenant administrator role, follow the guidelines for your organization to request that the tenant administrator approve your permissions request.

```
$ az ad app permission grant --id <appld> \ 1
--api 00000002-0000-0000-c000-000000000000
```

- 1 Replace **<appld>** with the **appld** parameter value for your service principal.

3.1.6. Supported Azure regions

The installation program dynamically generates the list of available Microsoft Azure regions based on your subscription. The following Azure regions were tested and validated in OpenShift Container Platform version 4.5.4:

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)

- **norwayeast** (Norway East)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

3.1.7. Next steps

- Install an OpenShift Container Platform cluster on Azure. You can [install a customized cluster](#) or [quickly install a cluster](#) with default options.

3.2. MANUALLY CREATING IAM FOR AZURE

3.2.1. Manually create IAM

The Cloud Credential Operator can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Run the OpenShift Container Platform installer to generate manifests:

```
$ openshift-install create manifests --dir=mycluster
```

2. Insert a config map into the manifests directory so that the Cloud Credential Operator is placed in manual mode:

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-credential-operator-config
```



```

namespace: openshift-cloud-credential-operator
annotations:
  release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF

```

3. Remove the **admin** credential secret created using your local cloud credentials. This removal prevents your **admin** credential from being stored in the cluster:

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. Obtain the OpenShift Container Platform release image your **openshift-install** binary is built to use:

```
$ bin/openshift-install version
```

Example output

```
release image quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64
```

5. Locate all **CredentialsRequest** objects in this release image that target the cloud you are deploying on:

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64 --to
./release-image
```

6. Locate the **CredentialsRequests** in the extracted file:

```
$ grep -l "apiVersion: cloudcredential.openshift.io" * | xargs cat
```



NOTE

In a future OpenShift Container Platform release, there will be a new **oc adm release** command to scan for the **CredentialsRequests** and display them.

This displays the details for each request. Remember to ignore any **CredentialsRequests** where the **spec.providerSpec.kind** does not match the cloud provider you are installing to.

Sample **CredentialsRequest** object

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-azure
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry

```

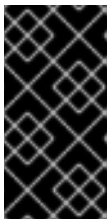
```

providerSpec:
  apiVersion: cloudcredential.openshift.io/v1
  kind: AzureProviderSpec
  roleBindings:
    - role: Contributor

```

7. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in each **request.spec.secretRef**. The format for the secret data varies for each cloud provider.
8. Proceed with cluster creation:

```
$ openshift-install create cluster --dir=mycluster
```



IMPORTANT

Before performing an upgrade, you might need to adjust your credentials if permissions have changed in the next release. In the future, the Cloud Credential Operator might prevent you from upgrading until you have indicated that you have addressed updated permissions.

3.2.2. Admin credentials root secret format

Each cloud provider uses a credentials root secret in the **kube-system** namespace by convention, which is then used to satisfy all credentials requests and create their respective secrets. This is done either by minting new credentials, *Mint Mode*, or by copying the credentials root secret, *Passthrough Mode*.

The format for the secret varies by cloud, and is also used for each **CredentialsRequest** secret.

Microsoft Azure secret format

```

apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: azure-credentials
stringData:
  azure_subscription_id: <SubscriptionID>
  azure_client_id: <ClientID>
  azure_client_secret: <ClientSecret>
  azure_tenant_id: <TenantID>
  azure_resource_prefix: <ResourcePrefix>
  azure_resourcegroup: <ResourceGroup>
  azure_region: <Region>

```

On Microsoft Azure, the credentials secret format includes two properties that must contain the cluster's infrastructure ID, generated randomly for each cluster installation. This value can be found after running create manifests:

```
$ cat .openshift_install_state.json | jq '.*installconfig.ClusterID'.InfraID' -r
```

Example output

```
mycluster-2mpcn
```

This value would be used in the secret data as follows:

```
azure_resource_prefix: mycluster-2mpcn
azure_resourcegroup: mycluster-2mpcn-rg
```

3.2.2.1. Upgrades

In a future release, improvements to the Cloud Credential Operator will prevent situations where a user might enter an upgrade that will fail because their manually maintained credentials have not been updated to match the **CredentialsRequest** objects in the upcoming release image.

3.2.3. Mint Mode

Mint Mode is supported for AWS, GCP, and Azure.

The default and recommended best practice for running OpenShift Container Platform is to run the installer with an administrator-level cloud credential. The **admin** credential is stored in the **kube-system** namespace, and then used by the Cloud Credential Operator to process the **CredentialsRequest** objects in the cluster and create new users for each with specific permissions.

The benefits of Mint Mode include:

- Each cluster component only has the permissions it requires.
- Automatic, on-going reconciliation for cloud credentials including upgrades, which might require additional credentials or permissions.

One drawback is that Mint Mode requires **admin** credential storage in a cluster **kube-system** secret.

3.3. INSTALLING A CLUSTER QUICKLY ON AZURE

In OpenShift Container Platform version 4.5, you can install a cluster on Microsoft Azure that uses the default configuration options.

3.3.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

3.3.2. Internet and Telemetry access for OpenShift Container Platform

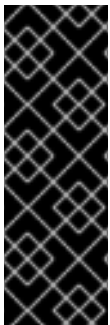
In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs

automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#) .

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

3.3.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

3.3.4. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

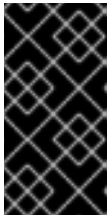
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

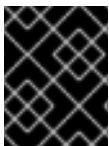
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.3.5. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

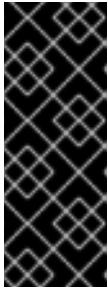
- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

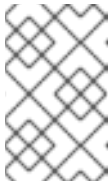


IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

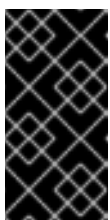
- a. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

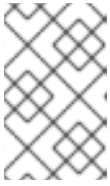
- b. Select **azure** as the platform to target.
- c. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantid** value in your account output.
 - **azure service principal client id** The value of the **appid** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- d. Select the region to deploy the cluster to.
- e. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- f. Enter a descriptive name for your cluster.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

- g. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.



NOTE

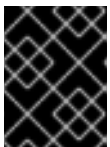
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3.3.6. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

3.3.6.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:


```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.3.6.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**. To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

3.3.6.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**. To check your **PATH**, open a terminal and execute the following command:

-

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.3.7. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

3.3.8. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

3.4. INSTALLING A CLUSTER ON AZURE WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a customized cluster on infrastructure that the installation program provisions on Microsoft Azure. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

3.4.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.

- [Configure an Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

3.4.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

3.4.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

3.4.4. Obtaining the installation program

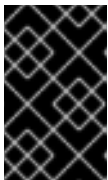
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

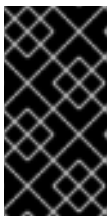
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.4.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Microsoft Azure.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

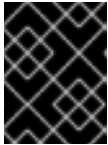
For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **azure** as the platform to target.
- iii. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantId** value in your account output.
 - **azure service principal client id** The value of the **appId** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- iv. Select the region to deploy the cluster to.
- v. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- vi. Enter a descriptive name for your cluster.

**IMPORTANT**

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

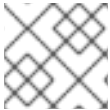
- vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

3.4.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

3.4.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 3.1. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


Parameter	Description	Values
-----------	-------------	--------


3.4.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 3.2. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>


3.4.5.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 3.3. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;"> <div style="background-color: black; width: 20px; height: 100px; margin-right: 10px;"></div> <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

3.4.5.1.4. Additional Azure configuration parameters

Additional Azure configuration parameters are described in the following table:

Table 3.4. Additional Azure parameters

Parameter	Description	Values
controlPlane.platform.azure.osDisk.diskSizeGB	The Azure disk size for the VM.	Integer that represents the size of the disk in GB. The minimum supported disk size is 1024 .
platform.azure.baseDomainResourceGroupName	The name of the resource group that contains the DNS zone for your base domain.	String, for example production_cluster .
platform.azure.region	The name of the Azure region that hosts your cluster.	Any valid region name, such as centralus .
platform.azure.zone	List of availability zones to place machines in. For high availability, specify at least two zones.	List of zones, for example ["1", "2", "3"] .
platform.azure.networkResourceGroupName	The name of the resource group that contains the existing VNet that you want to deploy your cluster to. This name cannot be the same as the platform.azure.baseDomainResourceGroupName .	String.
platform.azure.virtualNetwork	The name of the existing VNet that you want to deploy your cluster to.	String.

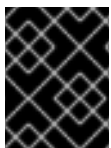
Parameter	Description	Values
platform.azure.controlPlaneSubnet	The name of the existing subnet in your VNet that you want to deploy your control plane machines to.	Valid CIDR, for example 10.0.0.0/16 .
platform.azure.computeSubnet	The name of the existing subnet in your VNet that you want to deploy your compute machines to.	Valid CIDR, for example 10.0.0.0/16 .

**NOTE**

You cannot customize [Azure Availability Zones](#) or [Use tags to organize your Azure resources](#) with an Azure cluster.

3.4.5.2. Sample customized install-config.yaml file for Azure

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

**IMPORTANT**

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 5
        type: Standard_D8s_v3
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:

```

```

name: test-cluster 10
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  azure:
    region: centralus 11
    baseDomainResourceGroupName: resource_group 12
pullSecret: '{"auths": ...}' 13
ifndef::openshift-origin
fips: false 14
sshKey: ssh-ed25519 AAAA... 15
endif::openshift-origin
ifdef::openshift-origin
sshKey: ssh-ed25519 AAAA... 16
endif::openshift-origin

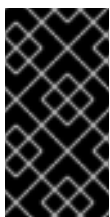
```

1 10 11 13 Required. The installation program prompts you for this value.

2 6 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 Whether to enable or disable simultaneous multithreading, or **hypertreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger virtual machine types, such as **Standard_D8s_v3**, for your machines if you disable simultaneous multithreading.

5 8 You can specify the size of the disk to use in GB. Minimum recommendation for master nodes is 1024 GB.

9 Specify a list of zones to deploy your machines to. For high availability, specify at least two zones.

12 Specify the name of the resource group that contains the DNS zone for your base domain.

14 **16** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container

15 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

3.4.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

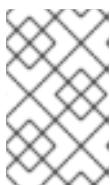
Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

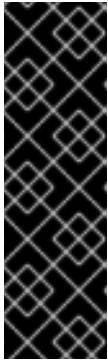
2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

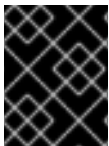
The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3.4.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

3.4.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.4.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

3.4.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.4.8. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

3.4.9. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

3.5. INSTALLING A CLUSTER ON AZURE WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster with a customized network configuration on infrastructure that the installation program provisions on Microsoft Azure. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

3.5.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

3.5.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

3.5.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

■

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

3.5.4. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.

- You need 500 MB of local disk space to download the installation program.

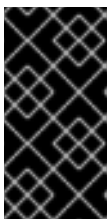
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.5.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Microsoft Azure.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

**IMPORTANT**

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

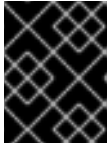
- ii. Select **azure** as the platform to target.
- iii. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantId** value in your account output.
 - **azure service principal client id** The value of the **appId** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- iv. Select the region to deploy the cluster to.
- v. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- vi. Enter a descriptive name for your cluster.

**IMPORTANT**

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

- vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

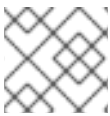


IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

3.5.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

3.5.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 3.5. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .

Parameter	Description	Values
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}.{{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


3.5.5.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 3.6. Network parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.



3.5.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 3.7. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hypertreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

3.5.5.1.4. Additional Azure configuration parameters

Additional Azure configuration parameters are described in the following table:

Table 3.8. Additional Azure parameters

Parameter	Description	Values
controlPlane.platform.azure.osDisk.diskSizeGB	The Azure disk size for the VM.	Integer that represents the size of the disk in GB. The minimum supported disk size is 1024 .

Parameter	Description	Values
platform.azure.baseDomainResourceGroupName	The name of the resource group that contains the DNS zone for your base domain.	String, for example production_cluster .
platform.azure.region	The name of the Azure region that hosts your cluster.	Any valid region name, such as centralus .
platform.azure.zone	List of availability zones to place machines in. For high availability, specify at least two zones.	List of zones, for example ["1", "2", "3"] .
platform.azure.networkResourceGroupName	The name of the resource group that contains the existing VNet that you want to deploy your cluster to. This name cannot be the same as the platform.azure.baseDomainResourceGroupName .	String.
platform.azure.virtualNetwork	The name of the existing VNet that you want to deploy your cluster to.	String.
platform.azure.controlPlaneSubnet	The name of the existing subnet in your VNet that you want to deploy your control plane machines to.	Valid CIDR, for example 10.0.0.0/16 .
platform.azure.computeSubnet	The name of the existing subnet in your VNet that you want to deploy your compute machines to.	Valid CIDR, for example 10.0.0.0/16 .



NOTE

You cannot customize [Azure Availability Zones](#) or [Use tags to organize your Azure resources](#) with an Azure cluster.



IMPORTANT

The Open Virtual Networking (OVN) Kubernetes network plug-in is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of the OVN Technology Preview, see <https://access.redhat.com/articles/4380121>.

3.5.5.2. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 3.9. Required network parameters

Parameter	Description	Value
networking.net workType	The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5. The OVNKubernetes plug-in is available as a Technology Preview in OpenShift Container Platform 4.5.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork[].cidr	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .
networking.clusterNetwork[].hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork[]	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.machineNetwork[].cidr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

3.5.5.3. Sample customized install-config.yaml file for Azure

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

■

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 5
        type: Standard_D8s_v3
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
      - "1"
      - "2"
      - "3"
    replicas: 5
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    region: centralus 12
    baseDomainResourceGroupName: resource_group 13
pullSecret: '{"auths": ...}' 14
ifndef::openshift-origin
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
endif::openshift-origin
ifndef::openshift-origin
sshKey: ssh-ed25519 AAAA... 17
endif::openshift-origin

```

1 10 12 14 Required. The installation program prompts you for this value.

2 6 11 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings.

To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

- 4 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger virtual machine types, such as **Standard_D8s_v3**, for your machines if you disable simultaneous multithreading.

- 5 8 You can specify the size of the disk to use in GB. Minimum recommendation for master nodes is 1024 GB.
- 9 Specify a list of zones to deploy your machines to. For high availability, specify at least two zones.
- 13 Specify the name of the resource group that contains the DNS zone for your base domain.
- 15 17 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 16 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

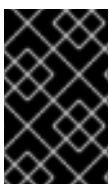


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

3.5.6. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster. Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of **kube-proxy** settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.

Procedure

1. Use the following command to create manifests:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1 For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, several network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

- 1 The parameters for the **spec** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.
5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

3.5.7. Cluster Network Operator configuration

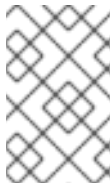
The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: ❶
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: ❷
  - 172.30.0.0/16
  defaultNetwork: ❸
  ...
  kubeProxyConfig: ❹
  iptablesSyncPeriod: 30s ❺
  proxyArguments:
    iptables-min-sync-period: ❻
    - 0s
```

- ❶ ❷ Specified in the **install-config.yaml** file.
- ❸ Configures the default Container Network Interface (CNI) network provider for the cluster network.
- ❹ The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.
- ❺ The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.

**NOTE**

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

- 6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#).

3.5.7.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**.

- 5 The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for VXLAN, since both SDNs use the same default VXLAN port number.

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

3.5.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider

The following YAML object describes the configuration parameters for the OVN-Kubernetes default CNI network provider.

```
defaultNetwork:
  type: OVNKubernetes 1
  ovnKubernetesConfig: 2
    mtu: 1400 3
    genevePort: 6081 4
```

- 1** Specified in the **install-config.yaml** file.
- 2** Specify only if you want to override part of the OVN-Kubernetes configuration.
- 3** The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **100** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1400**.

- 4** The UDP port for the Geneve overlay network.

3.5.7.3. Cluster Network Operator example configuration

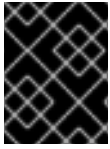
A complete CR object for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
  proxyArguments:
    iptables-min-sync-period:
      - 0s
```

3.5.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrap** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3.5.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

3.5.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.5.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

-

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

3.5.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.5.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

3.5.11. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .

3.6. INSTALLING A CLUSTER ON AZURE INTO AN EXISTING VNET

In OpenShift Container Platform version 4.5, you can install a cluster into an existing Azure Virtual Network (VNet) on Microsoft Azure. The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

3.6.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

3.6.2. About reusing a VNet for your OpenShift Container Platform cluster

In OpenShift Container Platform 4.5, you can deploy a cluster into an existing Azure Virtual Network (VNet) in Microsoft Azure. If you do, you must also use existing subnets within the VNet and routing rules.

By deploying OpenShift Container Platform into an existing Azure VNet, you might be able to avoid service limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. This is a good option to use if you cannot obtain the infrastructure creation permissions that are required to create the VNet.



IMPORTANT

The use of an existing VNet requires the use of the updated Azure Private DNS (preview) feature. See [Announcing Preview Refresh for Azure DNS Private Zones](#) for more information about the limitations of this feature.

3.6.2.1. Requirements for using your VNet

When you deploy a cluster by using an existing VNet, you must perform additional network configuration before you install the cluster. In installer-provisioned infrastructure clusters, the installer usually creates the following components, but it does not create them when you install into an existing VNet:

- Subnets
- Route tables
- VNets
- Network Security Groups

If you use a custom VNet, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VNet options like DHCP, so you must do so before you install the cluster.

The cluster must be able to access the resource group that contains the existing VNet and subnets. While all of the resources that the cluster creates are placed in a separate resource group that it creates, some network resources are used from a separate group. Some cluster Operators must be able to access resources in both resource groups. For example, the Machine API controller attaches NICS for the virtual machines that it creates to subnets from the networking resource group.

Your VNet must meet the following characteristics:

- The VNet's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VNet and its subnets must belong to the same resource group, and the subnets must be configured to use Azure-assigned DHCP IP addresses instead of static IP addresses.

You must provide two subnets within your VNet, one for the control plane machines and one for the compute machines. Because Azure distributes machines in different availability zones within the region that you specify, your cluster will have high availability by default.

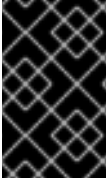
To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide two private subnets, one for the control plane machines and one for the compute machines.
- The subnet CIDRs belong to the machine CIDR that you specified. Machines are not provisioned in availability zones that you do not provide private subnets for. If required, the installation program creates public load balancers that manage the control plane and worker nodes, and Azure allocates a public IP address to them.

If you destroy a cluster that uses an existing VNet, the VNet is not deleted.

3.6.2.1.1. Network security group requirements

The network security groups for the subnets that host the compute and control plane machines require specific access to ensure that the cluster communication is correct. You must create rules to allow access to the required cluster communication ports.



IMPORTANT

The network security group rules must be in place before you install the cluster. If you attempt to install a cluster without the required access, the installation program cannot reach the Azure APIs, and installation fails.

Table 3.10. Required ports

Port	Description	Control plane	Compute
80	Allows HTTP traffic		x
443	Allows HTTPS traffic		x
6443	Allows communication to the control plane machines	x	
22623	Allows communication to the machine config server	x	



NOTE

Since cluster components do not modify the user-provided network security groups, which the Kubernetes controllers update, a pseudo-network security group is created for the Kubernetes controller to modify without impacting the rest of the environment.

3.6.2.2. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resources in your clouds than others. For example, you might be able to create application-specific items, like instances, storage, and load balancers, but not networking-related components such as VNets, subnet, or ingress rules.

The Azure credentials that you use when you create your cluster do not need the networking permissions that are required to make VNets and core networking components within the VNet, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as load balancers, security groups, storage accounts, and nodes.

3.6.2.3. Isolation between clusters

Because the cluster is unable to modify network security groups in an existing subnet, there is no way to isolate clusters from each other on the VNet.

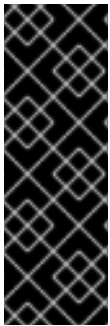
3.6.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

3.6.4. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

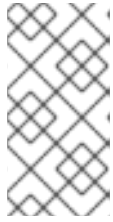
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

3.6.5. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.6.6. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Microsoft Azure.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **azure** as the platform to target.
- iii. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantId** value in your account output.
 - **azure service principal client id** The value of the **appId** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- iv. Select the region to deploy the cluster to.
- v. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- vi. Enter a descriptive name for your cluster.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

- vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

3.6.6.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

3.6.6.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 3.11. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .

Parameter	Description	Values
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


3.6.6.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 3.12. Network parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>


3.6.6.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 3.13. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hypert hreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platfor m	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replica s	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

3.6.6.1.4. Additional Azure configuration parameters

Additional Azure configuration parameters are described in the following table:

Table 3.14. Additional Azure parameters

Parameter	Description	Values
controlPlane.platform.azure.osDisk.diskSizeGB	The Azure disk size for the VM.	Integer that represents the size of the disk in GB. The minimum supported disk size is 1024 .
platform.azure.baseDomainResourceGroupName	The name of the resource group that contains the DNS zone for your base domain.	String, for example production_cluster .
platform.azure.region	The name of the Azure region that hosts your cluster.	Any valid region name, such as centralus .
platform.azure.zone	List of availability zones to place machines in. For high availability, specify at least two zones.	List of zones, for example ["1", "2", "3"] .
platform.azure.networkResourceGroupName	The name of the resource group that contains the existing VNet that you want to deploy your cluster to. This name cannot be the same as the platform.azure.baseDomainResourceGroupName .	String.
platform.azure.virtualNetwork	The name of the existing VNet that you want to deploy your cluster to.	String.
platform.azure.controlPlaneSubnet	The name of the existing subnet in your VNet that you want to deploy your control plane machines to.	Valid CIDR, for example 10.0.0.0/16 .
platform.azure.computeSubnet	The name of the existing subnet in your VNet that you want to deploy your compute machines to.	Valid CIDR, for example 10.0.0.0/16 .



NOTE

You cannot customize [Azure Availability Zones](#) or [Use tags to organize your Azure resources](#) with an Azure cluster.

3.6.6.2. Sample customized install-config.yaml file for Azure

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 5
        type: Standard_D8s_v3
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
      - "1"
      - "2"
      - "3"
    replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  azure:
    region: centralus 11
    baseDomainResourceGroupName: resource_group 12
    networkResourceGroupName: vnet_resource_group 13
    virtualNetwork: vnet 14
    controlPlaneSubnet: control_plane_subnet 15
    computeSubnet: compute_subnet 16
pullSecret: '{"auths": ...}' 17
fips: false 18
sshKey: ssh-ed25519 AAAA... 19

```

1 10 11 17 Required. The installation program prompts you for this value.

- 2 6 If you do not provide these parameters and values, the installation program provides the default value.
- 3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 4 Whether to enable or disable simultaneous multithreading, or **hypertreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger virtual machine types, such as **Standard_D8s_v3**, for your machines if you disable simultaneous multithreading.

- 5 8 You can specify the size of the disk to use in GB. Minimum recommendation for master nodes is 1024 GB.
- 9 Specify a list of zones to deploy your machines to. For high availability, specify at least two zones.
- 12 Specify the name of the resource group that contains the DNS zone for your base domain.
- 13 If you use an existing VNet, specify the name of the resource group that contains it.
- 14 If you use an existing VNet, specify its name.
- 15 If you use an existing VNet, specify the name of the subnet to host the control plane machines.
- 16 If you use an existing VNet, specify the name of the subnet to host the compute machines.
- 18 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 19 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

3.6.6.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

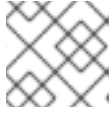
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
```

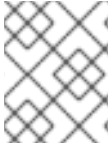
- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle**

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

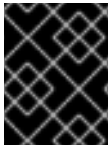
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

3.6.7. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

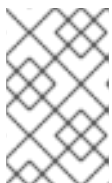
- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1** For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

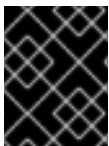


IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3.6.8. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

3.6.8.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.6.8.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

3.6.8.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.6.9. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

3.6.10. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

3.7. INSTALLING A PRIVATE CLUSTER ON AZURE

In OpenShift Container Platform version 4.5, you can install a private cluster into an existing Azure Virtual Network (VNet) on Microsoft Azure. The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

3.7.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an Azure account](#) to host the cluster and determine the tested and validated region to deploy the cluster to.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#) . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

3.7.2. Private clusters

If your environment does not require an external Internet connection, you can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the Internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

To deploy a private cluster, you must use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.

Additionally, you must deploy a private cluster from a machine that has access the API services for the cloud you provision to, the hosts on the network that you provision, and to the internet to obtain installation media. You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

3.7.2.1. Private clusters in Azure

To create a private cluster on Microsoft Azure, you must provide an existing private VNet and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for only internal traffic.

Depending how your network connects to the private VNET, you might need to use a DNS forwarder in order to resolve the cluster's private DNS records. The cluster's machines use **168.63.129.16** internally for DNS resolution. For more information, see [What is Azure Private DNS?](#) and [What is IP address 168.63.129.16?](#) in the Azure documentation.

The cluster still requires access to Internet to access the Azure APIs.

The following items are not required or created when you install a private cluster:

- A **BaseDomainResourceGroup**, since the cluster does not create public records
- Public IP addresses
- Public DNS records
- Public endpoints

The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

3.7.2.1.1. Limitations

Private clusters on Azure are subject to only the limitations that are associated with the use of an existing VNet.

3.7.3. About reusing a VNet for your OpenShift Container Platform cluster

In OpenShift Container Platform 4.5, you can deploy a cluster into an existing Azure Virtual Network (VNet) in Microsoft Azure. If you do, you must also use existing subnets within the VNet and routing rules.

By deploying OpenShift Container Platform into an existing Azure VNet, you might be able to avoid service limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. This is a good option to use if you cannot obtain the infrastructure creation permissions that are required to create the VNet.



IMPORTANT

The use of an existing VNet requires the use of the updated Azure Private DNS (preview) feature. See [Announcing Preview Refresh for Azure DNS Private Zones](#) for more information about the limitations of this feature.

3.7.3.1. Requirements for using your VNet

When you deploy a cluster by using an existing VNet, you must perform additional network configuration before you install the cluster. In installer-provisioned infrastructure clusters, the installer usually creates the following components, but it does not create them when you install into an existing VNet:

- Subnets
- Route tables
- VNets
- Network Security Groups

If you use a custom VNet, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VNet options like DHCP, so you must do so before you install the cluster.

The cluster must be able to access the resource group that contains the existing VNet and subnets. While all of the resources that the cluster creates are placed in a separate resource group that it creates, some network resources are used from a separate group. Some cluster Operators must be able to access resources in both resource groups. For example, the Machine API controller attaches NICs for the virtual machines that it creates to subnets from the networking resource group.

Your VNet must meet the following characteristics:

- The VNet's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines.
- The VNet and its subnets must belong to the same resource group, and the subnets must be configured to use Azure-assigned DHCP IP addresses instead of static IP addresses.

You must provide two subnets within your VNet, one for the control plane machines and one for the compute machines. Because Azure distributes machines in different availability zones within the region that you specify, your cluster will have high availability by default.

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

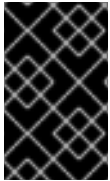
- All the subnets that you specify exist.
- You provide two private subnets, one for the control plane machines and one for the compute machines.

- The subnet CIDRs belong to the machine CIDR that you specified. Machines are not provisioned in availability zones that you do not provide private subnets for. If required, the installation program creates public load balancers that manage the control plane and worker nodes, and Azure allocates a public IP address to them.

If you destroy a cluster that uses an existing VNet, the VNet is not deleted.

3.7.3.1.1. Network security group requirements

The network security groups for the subnets that host the compute and control plane machines require specific access to ensure that the cluster communication is correct. You must create rules to allow access to the required cluster communication ports.



IMPORTANT

The network security group rules must be in place before you install the cluster. If you attempt to install a cluster without the required access, the installation program cannot reach the Azure APIs, and installation fails.

Table 3.15. Required ports

Port	Description	Control plane	Compute
80	Allows HTTP traffic		x
443	Allows HTTPS traffic		x
6443	Allows communication to the control plane machines	x	
22623	Allows communication to the machine config server	x	



NOTE

Since cluster components do not modify the user-provided network security groups, which the Kubernetes controllers update, a pseudo-network security group is created for the Kubernetes controller to modify without impacting the rest of the environment.

3.7.3.2. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resources in your clouds than others. For example, you might be able to create application-specific items, like instances, storage, and load balancers, but not networking-related components such as VNets, subnet, or ingress rules.

The Azure credentials that you use when you create your cluster do not need the networking permissions that are required to make VNets and core networking components within the VNet, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as load balancers, security groups, storage accounts, and nodes.

3.7.3.3. Isolation between clusters

Because the cluster is unable to modify network security groups in an existing subnet, there is no way to isolate clusters from each other on the VNet.

3.7.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

3.7.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

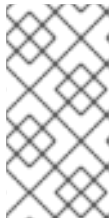
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

3.7.6. Obtaining the installation program

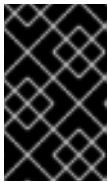
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

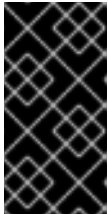
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.7.7. Manually creating the installation configuration file

For installations of a private OpenShift Container Platform cluster that are only accessible from an internal network and are not visible to the Internet, you must manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

**IMPORTANT**

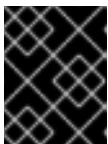
You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

**NOTE**

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

3.7.7.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

3.7.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 3.16. Required parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object


Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


3.7.7.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 3.17. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object <div style="display: flex; align-items: center; margin-top: 10px;">  <div> <p>NOTE</p> <p>You cannot modify parameters specified by the networking object after installation.</p> </div> </div>
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	<p>The IP address blocks for pods.</p> <p>The default value is 10.128.0.0/14 with a host prefix of /23.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


3.7.7.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 3.18. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div data-bbox="485 1160 592 1476" style="background-color: black; width: 65px; height: 141px; margin-bottom: 10px;"></div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

3.7.7.1.4. Additional Azure configuration parameters

Additional Azure configuration parameters are described in the following table:

Table 3.19. Additional Azure parameters

Parameter	Description	Values
controlPlane.platform.azure.osDisk.diskSizeGB	The Azure disk size for the VM.	Integer that represents the size of the disk in GB. The minimum supported disk size is 1024 .
platform.azure.baseDomainResourceGroupName	The name of the resource group that contains the DNS zone for your base domain.	String, for example production_cluster .
platform.azure.region	The name of the Azure region that hosts your cluster.	Any valid region name, such as centralus .
platform.azure.zone	List of availability zones to place machines in. For high availability, specify at least two zones.	List of zones, for example ["1", "2", "3"] .
platform.azure.networkResourceGroupName	The name of the resource group that contains the existing VNet that you want to deploy your cluster to. This name cannot be the same as the platform.azure.baseDomainResourceGroupName .	String.
platform.azure.virtualNetwork	The name of the existing VNet that you want to deploy your cluster to.	String.

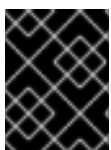
Parameter	Description	Values
platform.azure.controlPlaneSubnet	The name of the existing subnet in your VNet that you want to deploy your control plane machines to.	Valid CIDR, for example 10.0.0.0/16 .
platform.azure.computeSubnet	The name of the existing subnet in your VNet that you want to deploy your compute machines to.	Valid CIDR, for example 10.0.0.0/16 .

**NOTE**

You cannot customize [Azure Availability Zones](#) or [Use tags to organize your Azure resources](#) with an Azure cluster.

3.7.7.2. Sample customized install-config.yaml file for Azure

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

**IMPORTANT**

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 5
        type: Standard_D8s_v3
      replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:

```

```

name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    region: centralus 11
    baseDomainResourceGroupName: resource_group 12
    networkResourceGroupName: vnet_resource_group 13
    virtualNetwork: vnet 14
    controlPlaneSubnet: control_plane_subnet 15
    computeSubnet: compute_subnet 16
  pullSecret: '{"auths": ...}' 17
  fips: false 18
  sshKey: ssh-ed25519 AAAA... 19
  publish: Internal 20

```

1 10 11 17 Required. The installation program prompts you for this value.

2 6 If you do not provide these parameters and values, the installation program provides the default value.

3 7 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

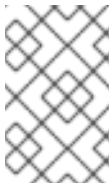
If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger virtual machine types, such as **Standard_D8s_v3**, for your machines if you disable simultaneous multithreading.

5 8 You can specify the size of the disk to use in GB. Minimum recommendation for master nodes is 1024 GB.

9 Specify a list of zones to deploy your machines to. For high availability, specify at least two zones.

12 Specify the name of the resource group that contains the DNS zone for your base domain.

- 13 If you use an existing VNet, specify the name of the resource group that contains it.
- 14 If you use an existing VNet, specify its name.
- 15 If you use an existing VNet, specify the name of the subnet to host the control plane machines.
- 16 If you use an existing VNet, specify the name of the subnet to host the compute machines.
- 18 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 19 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 20 How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the Internet. The default value is **External**.

3.7.7.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

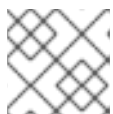
```
apiVersion: v1
```

```

baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

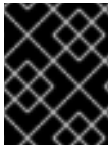


NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

3.7.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

1 For **<installation_directory>**, specify the

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

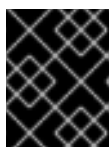
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

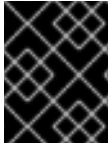


IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3.7.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

3.7.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.7.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**. To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

3.7.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.7.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

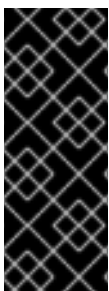
3.7.11. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .

3.8. INSTALLING A CLUSTER ON AZURE USING ARM TEMPLATES

In OpenShift Container Platform version 4.5, you can install a cluster on Microsoft Azure by using infrastructure that you provide.

Several [Azure Resource Manager](#) (ARM) templates are provided to assist in completing these steps or to help model your own.

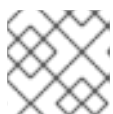


IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several ARM templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

3.8.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure an Azure account](#) to host the cluster.
- Download the Azure CLI and install it on your computer. See [Install the Azure CLI](#) in the Azure documentation. The documentation below was last tested using version **2.2.0** of the Azure CLI. Azure CLI commands might perform differently based on the version you use.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#) . Manual mode can also be used in environments where the cloud IAM APIs are not reachable.



NOTE

Be sure to also review this site list if you are configuring a proxy.

3.8.2. Internet and Telemetry access for OpenShift Container Platform

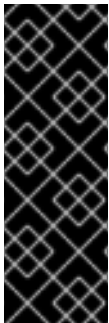
In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The

Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

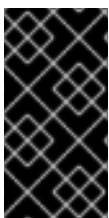


IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

3.8.3. Configuring your Azure project

Before you can install OpenShift Container Platform, you must configure an Azure project to host it.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

3.8.3.1. Azure account limits

The OpenShift Container Platform cluster uses a number of Microsoft Azure components, and the default [Azure subscription and service limits, quotas, and constraints](#) affect your ability to install OpenShift Container Platform clusters.



IMPORTANT

Default limits vary by offer category types, such as Free Trial and Pay-As-You-Go, and by series, such as Dv2, F, and G. For example, the default for Enterprise Agreement subscriptions is 350 cores.

Check the limits for your subscription type and if necessary, increase quota limits for your account before you install a default cluster on Azure.

The following table summarizes the Azure components whose limits can impact your ability to install and run OpenShift Container Platform clusters.

Component	Number of components required by default	Default Azure limit	Description
vCPU	40	20 per region	<p>A default cluster requires 40 vCPUs, so you must increase the account limit.</p> <p>By default, each cluster creates the following instances:</p> <ul style="list-style-type: none"> ● One bootstrap machine, which is removed after installation ● Three control plane machines ● Three compute machines <p>Because the bootstrap machine uses Standard_D4s_v3 machines, which use 4 vCPUs, the control plane machines use Standard_D8s_v3 virtual machines, which use 8 vCPUs, and the worker machines use Standard_D4s_v3 virtual machines, which use 4 vCPUs, a default cluster requires 40 vCPUs. The bootstrap node VM, which uses 4 vCPUs, is used only during installation.</p> <p>To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, you must further increase the vCPU limit for your account to ensure that your cluster can deploy the machines that you require.</p> <p>By default, the installation program distributes control plane and compute machines across all availability zones within a region. To ensure high availability for your cluster, select a region with at least three availability zones. If your region contains fewer than three availability zones, the installation program places more than one control plane machine in the available zones.</p>
VNet	1	1000 per region	Each default cluster requires one Virtual Network (VNet), which contains two subnets.
Network interfaces	6	65,536 per region	Each default cluster requires six network interfaces. If you create more machines or your deployed workloads create load balancers, your cluster uses more network interfaces.

Component	Number of components required by default	Default Azure limit	Description						
Network security groups	2	5000	<p>Each default cluster creates network security groups for each subnet in the VNet. The default cluster creates network security groups for the control plane and for the compute node subnets:</p> <table border="1"> <tr> <td>control plane</td> <td>Allows the control plane machines to be reached on port 6443 from anywhere</td> </tr> <tr> <td>node</td> <td>Allows worker nodes to be reached from the Internet on ports 80 and 443</td> </tr> </table>	control plane	Allows the control plane machines to be reached on port 6443 from anywhere	node	Allows worker nodes to be reached from the Internet on ports 80 and 443		
control plane	Allows the control plane machines to be reached on port 6443 from anywhere								
node	Allows worker nodes to be reached from the Internet on ports 80 and 443								
Network load balancers	3	1000 per region	<p>Each cluster creates the following load balancers:</p> <table border="1"> <tr> <td>default</td> <td>Public IP address that load balances requests to ports 80 and 443 across worker machines</td> </tr> <tr> <td>internal</td> <td>Private IP address that load balances requests to ports 6443 and 22623 across control plane machines</td> </tr> <tr> <td>external</td> <td>Public IP address that load balances requests to port 6443 across control plane machines</td> </tr> </table> <p>If your applications create more Kubernetes LoadBalancer service objects, your cluster uses more load balancers.</p>	default	Public IP address that load balances requests to ports 80 and 443 across worker machines	internal	Private IP address that load balances requests to ports 6443 and 22623 across control plane machines	external	Public IP address that load balances requests to port 6443 across control plane machines
default	Public IP address that load balances requests to ports 80 and 443 across worker machines								
internal	Private IP address that load balances requests to ports 6443 and 22623 across control plane machines								
external	Public IP address that load balances requests to port 6443 across control plane machines								
Public IP addresses	3		Each of the two public load balancers uses a public IP address. The bootstrap machine also uses a public IP address so that you can SSH into the machine to troubleshoot issues during installation. The IP address for the bootstrap node is used only during installation.						
Private IP addresses	7		The internal load balancer, each of the three control plane machines, and each of the three worker machines each use a private IP address.						

3.8.3.2. Configuring a public DNS zone in Azure

To install OpenShift Container Platform, the Microsoft Azure account you use must have a dedicated public hosted DNS zone in your account. This zone must be authoritative for the domain. This service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through Azure or another source.



NOTE

For more information about purchasing domains through Azure, see [Buy a custom domain name for Azure App Service](#) in the Azure documentation.

2. If you are using an existing domain and registrar, migrate its DNS to Azure. See [Migrate an active DNS name to Azure App Service](#) in the Azure documentation.
3. Configure DNS for your domain. Follow the steps in the [Tutorial: Host your domain in Azure DNS](#) in the Azure documentation to create a public hosted zone for your domain or subdomain, extract the new authoritative name servers, and update the registrar records for the name servers that your domain uses.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
4. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

You can view Azure's DNS solution by visiting this [example for creating DNS zones](#).

3.8.3.3. Increasing Azure account limits

To increase an account limit, file a support request on the Azure portal.



NOTE

You can increase only one type of quota per support request.

Procedure

1. From the Azure portal, click **Help + support** in the lower left corner.
2. Click **New support request** and then select the required values:
 - a. From the **Issue type** list, select **Service and subscription limits (quotas)**
 - b. From the **Subscription** list, select the subscription to modify.
 - c. From the **Quota type** list, select the quota to increase. For example, select **Compute-VM (cores-vCPUs) subscription limit increases** to increase the number of vCPUs, which is required to install a cluster.
 - d. Click **Next: Solutions**.
3. On the **Problem Details** page, provide the required information for your quota increase:

- a. Click **Provide details** and provide the required details in the **Quota details** window.
 - b. In the SUPPORT METHOD and CONTACT INFO sections, provide the issue severity and your contact details.
4. Click **Next: Review + create** and then click **Create**.

3.8.3.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

3.8.3.5. Required Azure roles

Your Microsoft Azure account must have the following roles for the subscription that you use:

- **User Access Administrator**

To set roles on the Azure portal, see the [Manage access to Azure resources using RBAC and the Azure portal](#) in the Azure documentation.

3.8.3.6. Creating a service principal

Because OpenShift Container Platform and its installation program must create Microsoft Azure resources through Azure Resource Manager, you must create a service principal to represent it.

Prerequisites

- Install or update the [Azure CLI](#).
- Install the **jq** package.
- Your Azure account has the required roles for the subscription that you use.

Procedure

1. Log in to the Azure CLI:

```
$ az login
```

Log in to Azure in the web console by using your credentials.

2. If your Azure account uses subscriptions, ensure that you are using the right subscription.
 - a. View the list of available accounts and record the **tenantId** value for the subscription you want to use for your cluster:

```
$ az account list --refresh
```

Example output

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

- b. View your active account details and confirm that the **tenantId** value matches the subscription you want to use:

```
$ az account show
```

Example output

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", 1
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

- 1** Ensure that the value of the **tenantId** parameter is the UUID of the correct subscription.

- c. If you are not using the right subscription, change the active subscription:

```
$ az account set -s <id> 1
```

- 1** Substitute the value of the **id** for the subscription that you want to use for **<id>**.

- d. If you changed the active subscription, display your account information again:

```
$ az account show
```

Example output

```
{
```

```

"environmentName": "AzureCloud",
"id": "33212d16-bdf6-45cb-b038-f6565b61edda",
"isDefault": true,
"name": "Subscription Name",
"state": "Enabled",
"tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
"user": {
  "name": "you@example.com",
  "type": "user"
}
}

```

- Record the values of the **tenantId** and **id** parameters from the previous output. You need these values during OpenShift Container Platform installation.
- Create the service principal for your account:

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
```

- Replace **<service_principal>** with the name to assign to the service principal.

Example output

```

Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the
required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
Retrying role assignment creation: 4/36
{
  "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
  "displayName": "<service_principal>",
  "name": "http://<service_principal>",
  "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
}

```

- Record the values of the **appId** and **password** parameters from the previous output. You need these values during OpenShift Container Platform installation.
- Grant additional permissions to the service principal. The service principal requires the legacy **Azure Active Directory Graph** → **Application.ReadWrite.OwnedBy** permission and the **User Access Administrator** role for the cluster to assign credentials for its components.
 - To assign the **User Access Administrator** role, run the following command:

```

$ az role assignment create --role "User Access Administrator" \
  --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \
  | jq '[0].objectId' -r) 1

```

- Replace **<appId>** with the **appId** parameter value for your service principal.

- To assign the **Azure Active Directory Graph** permission, run the following command:

```
$ az ad app permission add --id <appld> \ 1  
--api 00000002-0000-0000-c000-000000000000 \  
--api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
```

- 1** Replace **<appld>** with the **appld** parameter value for your service principal.

Example output

```
Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api  
00000002-0000-0000-c000-000000000000" is needed to make the change effective
```

For more information about the specific permissions that you grant with this command, see the [GUID Table for Windows Azure Active Directory Permissions](#) .

- c. Approve the permissions request. If your account does not have the Azure Active Directory tenant administrator role, follow the guidelines for your organization to request that the tenant administrator approve your permissions request.

```
$ az ad app permission grant --id <appld> \ 1  
--api 00000002-0000-0000-c000-000000000000
```

- 1** Replace **<appld>** with the **appld** parameter value for your service principal.

3.8.3.7. Supported Azure regions

The installation program dynamically generates the list of available Microsoft Azure regions based on your subscription. The following Azure regions were tested and validated in OpenShift Container Platform version 4.5.4:

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)
- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)

- **germanywestcentral** (Germany West Central)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

3.8.4. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

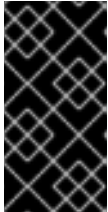
1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

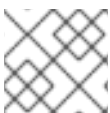
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

3.8.5. Generating an SSH private key and adding it to the agent

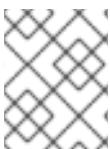
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

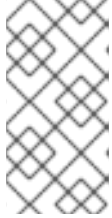
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```


- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

3.8.6. Creating the installation files for Azure

To install OpenShift Container Platform on Microsoft Azure using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files.

3.8.6.1. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Microsoft Azure.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **azure** as the platform to target.
- iii. If you do not have a Microsoft Azure profile stored on your computer, specify the following Azure parameter values for your subscription and service principal:
 - **azure subscription id** The subscription ID to use for the cluster. Specify the **id** value in your account output.
 - **azure tenant id** The tenant ID. Specify the **tenantId** value in your account output.
 - **azure service principal client id** The value of the **appId** parameter for the service principal.
 - **azure service principal client secret** The value of the **password** parameter for the service principal.
- iv. Select the region to deploy the cluster to.

- v. Select the base domain to deploy the cluster to. The base domain corresponds to the Azure DNS Zone that you created for your cluster.
- vi. Enter a descriptive name for your cluster.



IMPORTANT

All Azure resources that are available through public endpoints are subject to resource name restrictions, and you cannot create resources that use certain terms. For a list of terms that Azure restricts, see [Resolve reserved resource name errors](#) in the Azure documentation.

- vii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

3.8.6.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```

apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.



NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

3.8.6.3. Exporting common variables for ARM templates

You must export a common set of variables that are used with the provided Azure Resource Manager (ARM) templates used to assist in completing a user-provided infrastructure install on Microsoft Azure.



NOTE

Specific ARM templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Export common variables found in the **install-config.yaml** to be used by the provided ARM templates:

```
$ export CLUSTER_NAME=<cluster_name> 1
$ export AZURE_REGION=<azure_region> 2
$ export SSH_KEY=<ssh_key> 3
$ export BASE_DOMAIN=<base_domain> 4
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> 5
```

- 1 The value of the **.metadata.name** attribute from the **install-config.yaml** file.
- 2 The region to deploy the cluster into, for example **centralus**. This is the value of the **.platform.azure.region** attribute from the **install-config.yaml** file.
- 3 The SSH RSA public key file as a string. You must enclose the SSH key in quotes since it contains spaces. This is the value of the **.sshKey** attribute from the **install-config.yaml** file.
- 4 The base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster. This is the value of the **.baseDomain** attribute from the **install-config.yaml** file.
- 5 The resource group where the public DNS zone exists. This is the value of the **.platform.azure.baseDomainResourceGroupName** attribute from the **install-config.yaml** file.

For example:

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. Export the kubeadmin credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

3.8.6.4. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
 - b. Locate the `mastersSchedulable` parameter and set its value to **False**.
 - c. Save and exit the file.
5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}
```

- ❶ ❷ Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

6. When configuring Azure on user-provisioned infrastructure, you must export some common variables defined in the manifest files to use later in the Azure Resource Manager (ARM) templates:
 - a. Export the infrastructure ID by using the following command:

```
$ export INFRA_ID=<infra_id> ❶
```

- ❶ The OpenShift Container Platform cluster has been assigned an identifier (`INFRA_ID`) in the form of `<cluster_name>-<random_string>`. This will be used as the base name for most resources created using the provided ARM templates. This is the value of the `.status.infrastructureName` attribute from the `manifests/cluster-infrastructure-02-config.yml` file.

- b. Export the resource group by using the following command:

```
$ export RESOURCE_GROUP=<resource_group> ❶
```

- ❶ All resources created in this Azure deployment exists as part of a [resource group](#). The resource group name is also based on the `INFRA_ID`, in the form of `<cluster_name>-<random_string>-rg`. This is the value of the

`.status.platformStatus.azure.resourceGroupName` attribute from the `manifests/cluster-infrastructure-02-config.yml` file.

7. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

3.8.7. Creating the Azure resource group and identity

You must create a Microsoft Azure [resource group](#) and an identity for that resource group. These are both used during the installation of your OpenShift Container Platform cluster on Azure.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.

Procedure

1. Create the resource group in a supported Azure region:

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

2. Create an Azure identity for the resource group:

```
$ az identity create -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity
```

This is used to grant the required access to Operators in your cluster. For example, this allows the Ingress Operator to create a public IP and its load balancer. You must assign the Azure identity to a role.

3. Grant the Contributor role to the Azure identity:

a. Export the following variables required by the Azure role assignment:

```
$ export PRINCIPAL_ID=`az identity show -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity --query principalId --out tsv`
```



```
$ export RESOURCE_GROUP_ID=`az group show -g ${RESOURCE_GROUP} --query id --out tsv`
```

- b. Assign the Contributor role to the identity:

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role 'Contributor' --scope "${RESOURCE_GROUP_ID}"
```

3.8.8. Uploading the RHCOS cluster image and bootstrap Ignition config file

The Azure client does not support deployments based on files existing locally; therefore, you must copy and store the RHCOS virtual hard disk (VHD) cluster image and bootstrap Ignition config file in a storage container so they are accessible during deployment.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.

Procedure

1. Create an Azure storage account to store the VHD cluster image:

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```



WARNING

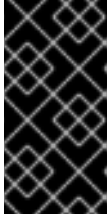
The Azure storage account name must be between 3 and 24 characters in length and use numbers and lower-case letters only. If your **CLUSTER_NAME** variable does not follow these restrictions, you must manually define the Azure storage account name. For more information on Azure storage account name restrictions, see [Resolve errors for storage account names](#) in the Azure documentation.

2. Export the storage account key as an environment variable:

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. Choose the RHCOS version to use and export the URL of its VHD to an environment variable:

```
$ export VHD_URL=`curl -s https://raw.githubusercontent.com/openshift/installer/release-4.5/data/data/rhcos.json | jq -r .azure.url`
```



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must specify an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

- Copy the chosen VHD to a blob:

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY}
```

```
$ az storage blob copy start --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} --destination-blob "rhcos.vhd" --destination-container vhd --source-uri "${VHD_URL}"
```

To track the progress of the VHD copy task, run this script:

```
status="unknown"
while [ "$status" != "success" ]
do
  status=`az storage blob show --container-name vhd --name "rhcos.vhd" --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -o tsv --query properties.copy.status`
  echo $status
done
```

- Create a blob storage container and upload the generated **bootstrap.ign** file:

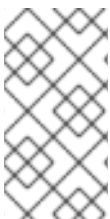
```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} --public-access blob
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

3.8.9. Example for creating DNS zones

DNS records are required for clusters that use user-provisioned infrastructure. You should choose the DNS strategy that fits your scenario.

For this example, [Azure's DNS solution](#) is used, so you will create a new public DNS zone for external (internet) visibility and a private DNS zone for internal cluster resolution.



NOTE

The public DNS zone is not required to exist in the same resource group as the cluster deployment and might already exist in your organization for the desired base domain. If that is the case, you can skip creating the public DNS zone; be sure the installation config you generated earlier reflects that scenario.

Prerequisites

- Configure an Azure account.

- Generate the Ignition config files for your cluster.

Procedure

1. Create the new public DNS zone in the resource group exported in the **BASE_DOMAIN_RESOURCE_GROUP** environment variable:

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

You can skip this step if you are using a public DNS zone that already exists.

2. Create the private DNS zone in the same resource group as the rest of this deployment:

```
$ az network private-dns zone create -g ${RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

You can learn more about [configuring a public DNS zone in Azure](#) by visiting that section.

3.8.10. Creating a VNet in Azure

You must create a virtual network (VNet) in Microsoft Azure for your OpenShift Container Platform cluster to use. You can customize the VNet to meet your requirements. One way to create the VNet is to modify the provided Azure Resource Manager (ARM) template.



NOTE

If you do not use the provided ARM template to create your Azure infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.

Procedure

1. Copy the template from the **ARM template for the VNet** section of this topic and save it as **01_vnet.json** in your cluster's installation directory. This template describes the VNet that your cluster requires.
2. Create the deployment by using the **az** CLI:

```
$ az deployment group create -g ${RESOURCE_GROUP} \
--template-file "<installation_directory>/01_vnet.json" \
--parameters baseName="${INFRA_ID}" 1
```

1 The base name to be used in resource names; this is usually the cluster's infrastructure ID.

3. Link the VNet template to the private DNS zone:

```
$ az network private-dns link vnet create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n ${INFRA_ID}-network-link -v "${INFRA_ID}-vnet"
-e false
```

3.8.10.1. ARM template for the VNet

You can use the following Azure Resource Manager (ARM) template to deploy the VNet that you need for your OpenShift Container Platform cluster:

Example 3.1. 01_vnet.json ARM template

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "addressPrefix" : "10.0.0.0/16",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetPrefix" : "10.0.0.0/24",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetPrefix" : "10.0.1.0/24",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/virtualNetworks",
      "name" : "[variables('virtualNetworkName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/networkSecurityGroups/', variables('clusterNsgName'))]"
      ],
      "properties" : {
        "addressSpace" : {
          "addressPrefixes" : [
            "[variables('addressPrefix')]"
          ]
        },
        "subnets" : [
          {
            "name" : "[variables('masterSubnetName')]",
            "properties" : {
              "addressPrefix" : "[variables('masterSubnetPrefix')]",
              "serviceEndpoints": [],

```

```

        "networkSecurityGroup" : {
            "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
        }
    },
    {
        "name" : "[variables('nodeSubnetName')]",
        "properties" : {
            "addressPrefix" : "[variables('nodeSubnetPrefix')]",
            "serviceEndpoints" : [],
            "networkSecurityGroup" : {
                "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
            }
        }
    }
]
},
{
    "type" : "Microsoft.Network/networkSecurityGroups",
    "name" : "[variables('clusterNsgName')]",
    "apiVersion" : "2018-10-01",
    "location" : "[variables('location')]",
    "properties" : {
        "securityRules" : [
            {
                "name" : "apiserver_in",
                "properties" : {
                    "protocol" : "Tcp",
                    "sourcePortRange" : "*",
                    "destinationPortRange" : "6443",
                    "sourceAddressPrefix" : "*",
                    "destinationAddressPrefix" : "*",
                    "access" : "Allow",
                    "priority" : 101,
                    "direction" : "Inbound"
                }
            }
        ]
    }
}
]
}
}

```

3.8.11. Deploying the RHCOS cluster image for the Azure infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Microsoft Azure for your OpenShift Container Platform nodes.

Prerequisites

- Configure an Azure account.

- Generate the Ignition config files for your cluster.
- Store the RHCOS virtual hard disk (VHD) cluster image in an Azure storage container.
- Store the bootstrap Ignition config file in an Azure storage container.

Procedure

1. Copy the template from the **ARM template for image storage** section of this topic and save it as **02_storage.json** in your cluster's installation directory. This template describes the image storage that your cluster requires.
2. Export the RHCOS VHD blob URL as a variable:

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. Deploy the cluster image:

```
$ az deployment group create -g ${RESOURCE_GROUP} \
--template-file "<installation_directory>/02_storage.json" \
--parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
--parameters baseName="${INFRA_ID}" 2
```

1 The blob URL of the RHCOS VHD to be used to create master and worker machines.

2 The base name to be used in resource names; this is usually the cluster's infrastructure ID.

3.8.11.1. ARM template for image storage

You can use the following Azure Resource Manager (ARM) template to deploy the stored Red Hat Enterprise Linux CoreOS (RHCOS) image that you need for your OpenShift Container Platform cluster:

Example 3.2. 02_storage.json ARM template

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vhdBlobURL" : {
      "type" : "string",
      "metadata" : {
        "description" : "URL pointing to the blob where the VHD to be used to create master and worker machines is located"
      }
    }
  }
}
```

```

},
"variables" : {
  "location" : "[resourceGroup().location]",
  "imageName" : "[concat(parameters('baseName'), '-image')]"
},
"resources" : [
  {
    "apiVersion" : "2018-06-01",
    "type": "Microsoft.Compute/images",
    "name": "[variables('imageName')]",
    "location" : "[variables('location')]",
    "properties": {
      "storageProfile": {
        "osDisk": {
          "osType": "Linux",
          "osState": "Generalized",
          "blobUri": "[parameters('vhdBlobURL')]",
          "storageAccountType": "Standard_LRS"
        }
      }
    }
  }
]
}

```

3.8.12. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 3.20. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve

Protocol	Port	Description
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 3.21. All machines to control plane

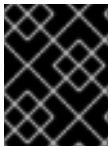
Protocol	Port	Description
TCP	6443	Kubernetes API

Table 3.22. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



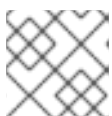
IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 3.23. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 3.24. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

3.8.13. Creating networking and load balancing components in Azure

You must configure networking and load balancing in Microsoft Azure for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Azure Resource Manager (ARM) template.

**NOTE**

If you do not use the provided ARM template to create your Azure infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.
- Create and configure a VNet and associated subnets in Azure.

Procedure

1. Copy the template from the **ARM template for the network and load balancers** section of this topic and save it as **03_infra.json** in your cluster's installation directory. This template describes the networking and load balancing objects that your cluster requires.
2. Create the deployment by using the **az** CLI:

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

- 1** The name of the private DNS zone.
- 2** The base name to be used in resource names; this is usually the cluster's infrastructure ID.

3. Create an **api** DNS record in the public zone for the API public load balancer. The **\${BASE_DOMAIN_RESOURCE_GROUP}** variable must point to the resource group where the public DNS zone exists.
 - a. Export the following variable:

```
$ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
```

- b. Create the DNS record in a new public zone:

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
```

- c. If you are adding the cluster to an existing public zone, you can create the DNS record in it instead:

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
```

3.8.13.1. ARM template for the network and load balancers

You can use the following Azure Resource Manager (ARM) template to deploy the networking objects and load balancers that you need for your OpenShift Container Platform cluster:

Example 3.3. 03_infra.json ARM template

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks', variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/', variables('masterSubnetName'))]",
    "masterPublicIpAddressName" : "[concat(parameters('baseName'), '-master-pip')]",
    "masterPublicIpAddressID" : "[resourceId('Microsoft.Network/publicIPAddresses', variables('masterPublicIpAddressName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "masterLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers', variables('masterLoadBalancerName'))]",
```

```

    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "internalLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('internalLoadBalancerName'))]",
    "skuName": "Standard"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('masterPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('masterPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('masterLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/publicIPAddresses/', variables('masterPublicIpAddressName'))]"
      ],
      "properties" : {
        "frontendIPConfigurations" : [
          {
            "name" : "public-lb-ip",
            "properties" : {
              "publicIPAddress" : {
                "id" : "[variables('masterPublicIpAddressID')]"
              }
            }
          }
        ],
        "backendAddressPools" : [
          {
            "name" : "public-lb-backend"
          }
        ],
        "loadBalancingRules" : [
          {
            "name" : "api-internal",
            "properties" : {
              "frontendIPConfiguration" : {
                "id" : "[concat(variables('masterLoadBalancerID'), '/frontendIPConfigurations/public-lb-
ip')]"
              },

```

```

    "backendAddressPool" : {
      "id" : "[concat(variables('masterLoadBalancerID'), '/backendAddressPools/public-lb-backend')]"
    },
    "protocol" : "Tcp",
    "loadDistribution" : "Default",
    "idleTimeoutInMinutes" : 30,
    "frontendPort" : 6443,
    "backendPort" : 6443,
    "probe" : {
      "id" : "[concat(variables('masterLoadBalancerID'), '/probes/api-internal-probe')]"
    }
  },
  ],
  "probes" : [
    {
      "name" : "api-internal-probe",
      "properties" : {
        "protocol" : "Https",
        "port" : 6443,
        "requestPath" : "/readyz",
        "intervalInSeconds" : 10,
        "numberOfProbes" : 3
      }
    }
  ]
},
{
  "apiVersion" : "2018-12-01",
  "type" : "Microsoft.Network/loadBalancers",
  "name" : "[variables('internalLoadBalancerName')]",
  "location" : "[variables('location')]",
  "sku" : {
    "name" : "[variables('skuName')]"
  },
  "properties" : {
    "frontendIPConfigurations" : [
      {
        "name" : "internal-lb-ip",
        "properties" : {
          "privateIPAllocationMethod" : "Dynamic",
          "subnet" : {
            "id" : "[variables('masterSubnetRef')]"
          },
          "privateIPAddressVersion" : "IPv4"
        }
      }
    ],
    "backendAddressPools" : [
      {
        "name" : "internal-lb-backend"
      }
    ],
    "loadBalancingRules" : [

```

```

    {
      "name" : "api-internal",
      "properties" : {
        "frontendIPConfiguration" : {
          "id" : "[concat(variables('internalLoadBalancerID'), 'frontendIPConfigurations/internal-lb-
ip')]"
        },
        "frontendPort" : 6443,
        "backendPort" : 6443,
        "enableFloatingIP" : false,
        "idleTimeoutInMinutes" : 30,
        "protocol" : "Tcp",
        "enableTcpReset" : false,
        "loadDistribution" : "Default",
        "backendAddressPool" : {
          "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
        },
        "probe" : {
          "id" : "[concat(variables('internalLoadBalancerID'), '/probes/api-internal-probe')]"
        }
      }
    },
    {
      "name" : "sint",
      "properties" : {
        "frontendIPConfiguration" : {
          "id" : "[concat(variables('internalLoadBalancerID'), 'frontendIPConfigurations/internal-lb-
ip')]"
        },
        "frontendPort" : 22623,
        "backendPort" : 22623,
        "enableFloatingIP" : false,
        "idleTimeoutInMinutes" : 30,
        "protocol" : "Tcp",
        "enableTcpReset" : false,
        "loadDistribution" : "Default",
        "backendAddressPool" : {
          "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
        },
        "probe" : {
          "id" : "[concat(variables('internalLoadBalancerID'), '/probes/sint-probe')]"
        }
      }
    }
  ],
  "probes" : [
    {
      "name" : "api-internal-probe",
      "properties" : {
        "protocol" : "Https",
        "port" : 6443,
        "requestPath" : "/readyz",
        "intervalInSeconds" : 10,
        "numberOfProbes" : 3
      }
    }
  ]
}

```

```

    }
  },
  {
    "name": "sint-probe",
    "properties": {
      "protocol": "Https",
      "port": 22623,
      "requestPath": "/healthz",
      "intervalInSeconds": 10,
      "numberOfProbes": 3
    }
  }
]
}
},
{
  "apiVersion": "2018-09-01",
  "type": "Microsoft.Network/privateDnsZones/A",
  "name": "[concat(parameters('privateDNSZoneName'), '/api')]",
  "location": "[variables('location')]",
  "dependsOn": [
    "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
  ],
  "properties": {
    "ttl": 60,
    "aRecords": [
      {
        "ipv4Address": "[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIPAddress]"
      }
    ]
  }
},
{
  "apiVersion": "2018-09-01",
  "type": "Microsoft.Network/privateDnsZones/A",
  "name": "[concat(parameters('privateDNSZoneName'), '/api-int')]",
  "location": "[variables('location')]",
  "dependsOn": [
    "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
  ],
  "properties": {
    "ttl": 60,
    "aRecords": [
      {
        "ipv4Address": "[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIPAddress]"
      }
    ]
  }
}
]
}

```

3.8.14. Creating the bootstrap machine in Azure

You must create the bootstrap machine in Microsoft Azure to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Azure Resource Manager (ARM) template.



NOTE

If you do not use the provided ARM template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.
- Create and configure a VNet and associated subnets in Azure.
- Create and configure networking and load balancers in Azure.
- Create control plane and compute roles.

Procedure

1. Copy the template from the **ARM template for the bootstrap machine** section of this topic and save it as **04_bootstrap.json** in your cluster's installation directory. This template describes the bootstrap machine that your cluster requires.
2. Export the following variables required by the bootstrap machine deployment:

```
$ export BOOTSTRAP_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c "files" -n "bootstrap.ign" -o tsv`
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "2.2.0" --arg url ${BOOTSTRAP_URL}
'{"ignition":{"version:$v,config:{replace:{source:$url}}}' | base64 -w0`
```

3. Create the deployment by using the **az** CLI:

```
$ az deployment group create -g ${RESOURCE_GROUP} \
--template-file "<installation_directory>/04_bootstrap.json" \
--parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
--parameters sshKeyData="${SSH_KEY}" \ 2
--parameters baseName="${INFRA_ID}" \ 3
```

- 1** The bootstrap Ignition content for the bootstrap cluster.
- 2** The SSH RSA public key file as a string.
- 3** The base name to be used in resource names; this is usually the cluster's infrastructure ID.

3.8.14.1. ARM template for the bootstrap machine

You can use the following Azure Resource Manager (ARM) template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 3.4. 04_bootstrap.json ARM template

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "bootstrapIgnition" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Bootstrap ignition content for the bootstrap cluster"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string."
      }
    },
    "bootstrapVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "allowedValues" : [
        "Standard_A2",
        "Standard_A3",
        "Standard_A4",
        "Standard_A5",
        "Standard_A6",
        "Standard_A7",
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",

```

```
"Standard_D11_v2",
"Standard_D12_v2",
"Standard_D13_v2",
"Standard_D14_v2",
"Standard_E2_v3",
"Standard_E4_v3",
"Standard_E8_v3",
"Standard_E16_v3",
"Standard_E32_v3",
"Standard_E64_v3",
"Standard_E2s_v3",
"Standard_E4s_v3",
"Standard_E8s_v3",
"Standard_E16s_v3",
"Standard_E32s_v3",
"Standard_E64s_v3",
"Standard_G1",
"Standard_G2",
"Standard_G3",
"Standard_G4",
"Standard_G5",
"Standard_DS2",
"Standard_DS3",
"Standard_DS4",
"Standard_DS11",
"Standard_DS12",
"Standard_DS13",
"Standard_DS14",
"Standard_DS2_v2",
"Standard_DS3_v2",
"Standard_DS4_v2",
"Standard_DS5_v2",
"Standard_DS11_v2",
"Standard_DS12_v2",
"Standard_DS13_v2",
"Standard_DS14_v2",
"Standard_GS1",
"Standard_GS2",
"Standard_GS3",
"Standard_GS4",
"Standard_GS5",
"Standard_D2s_v3",
"Standard_D4s_v3",
"Standard_D8s_v3"
],
"metadata" : {
  "description" : "The size of the Bootstrap Virtual Machine"
}
},
"variables" : {
  "location" : "[resourceGroup().location]",
  "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
  "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
  "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
```

```

"masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
"masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
"internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
"sshKeyPath" : "/home/core/.ssh/authorized_keys",
"identityName" : "[concat(parameters('baseName'), '-identity')]",
"vmName" : "[concat(parameters('baseName'), '-bootstrap')]",
"nicName" : "[concat(variables('vmName'), '-nic')]",
"imageName" : "[concat(parameters('baseName'), '-image')]",
"clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]",
"sshPublicIpAddressName" : "[concat(variables('vmName'), '-ssh-pip')]"
},
"resources" : [
{
"apiVersion" : "2018-12-01",
"type" : "Microsoft.Network/publicIPAddresses",
"name" : "[variables('sshPublicIpAddressName')]",
"location" : "[variables('location')]",
"sku": {
"name": "Standard"
},
"properties" : {
"publicIPAllocationMethod" : "Static",
"dnsSettings" : {
"domainNameLabel" : "[variables('sshPublicIpAddressName')]"
}
}
},
{
"apiVersion" : "2018-06-01",
"type" : "Microsoft.Network/networkInterfaces",
"name" : "[variables('nicName')]",
"location" : "[variables('location')]",
"dependsOn" : [
"[resourceId('Microsoft.Network/publicIPAddresses', variables('sshPublicIpAddressName'))]"
],
"properties" : {
"ipConfigurations" : [
{
"name" : "pipConfig",
"properties" : {
"privateIPAllocationMethod" : "Dynamic",
"publicIPAddress": {
"id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('sshPublicIpAddressName'))]"
},
"subnet" : {
"id" : "[variables('masterSubnetRef')]"
},
"loadBalancerBackendAddressPools" : [
{
"id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
},
{

```

```

        "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
    }
  ]
}
]
},
{
  "apiVersion" : "2018-06-01",
  "type" : "Microsoft.Compute/virtualMachines",
  "name" : "[variables('vmName')]",
  "location" : "[variables('location')]",
  "identity" : {
    "type" : "userAssigned",
    "userAssignedIdentities" : {
      "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
    }
  },
  "dependsOn" : [
    "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
  "properties" : {
    "hardwareProfile" : {
      "vmSize" : "[parameters('bootstrapVMSize')]"
    },
    "osProfile" : {
      "computerName" : "[variables('vmName')]",
      "adminUsername" : "core",
      "customData" : "[parameters('bootstrapIgnition')]",
      "linuxConfiguration" : {
        "disablePasswordAuthentication" : true,
        "ssh" : {
          "publicKeys" : [
            {
              "path" : "[variables('sshKeyPath')]",
              "keyData" : "[parameters('sshKeyData')]"
            }
          ]
        }
      }
    },
    "storageProfile" : {
      "imageReference": {
        "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
      },
      "osDisk" : {
        "name": "[concat(variables('vmName'),'_OSDisk')]",
        "osType" : "Linux",
        "createOption" : "FromImage",
        "managedDisk": {
          "storageAccountType": "Premium_LRS"
        }
      },

```

```

        "diskSizeGB" : 100
      }
    },
    "networkProfile" : {
      "networkInterfaces" : [
        {
          "id" : "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
        }
      ]
    }
  }
},
{
  "apiVersion" : "2018-06-01",
  "type" : "Microsoft.Network/networkSecurityGroups/securityRules",
  "name" : "[concat(variables('clusterNsgName'), '/bootstrap_ssh_in')]",
  "location" : "[variables('location')]",
  "dependsOn" : [
    "[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
  ],
  "properties" : {
    "protocol" : "Tcp",
    "sourcePortRange" : "*",
    "destinationPortRange" : "22",
    "sourceAddressPrefix" : "*",
    "destinationAddressPrefix" : "*",
    "access" : "Allow",
    "priority" : 100,
    "direction" : "Inbound"
  }
}
]
}

```

3.8.15. Creating the control plane machines in Azure

You must create the control plane machines in Microsoft Azure for your cluster to use. One way to create these machines is to modify the provided Azure Resource Manager (ARM) template.



NOTE

If you do not use the provided ARM template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.
- Create and configure a VNet and associated subnets in Azure.

- Create and configure networking and load balancers in Azure.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Copy the template from the **ARM template for control plane machines** section of this topic and save it as **05_masters.json** in your cluster's installation directory. This template describes the control plane machines that your cluster requires.
2. Export the following variable needed by the control plane machine deployment:

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64`
```

3. Create the deployment by using the **az** CLI:

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" 3
  --parameters baseName="${INFRA_ID}" 4
```

- 1** The Ignition content for the master nodes.
- 2** The SSH RSA public key file as a string.
- 3** The name of the private DNS zone to which the master nodes are attached.
- 4** The base name to be used in resource names; this is usually the cluster's infrastructure ID.

3.8.15.1. ARM template for control plane machines

You can use the following Azure Resource Manager (ARM) template to deploy the control plane machines that you need for your OpenShift Container Platform cluster:

Example 3.5. 05_masters.json ARM template

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "masterIgnition" : {
```

```

    "type" : "string",
    "metadata" : {
      "description" : "Ignition content for the master nodes"
    }
  },
  "numberOfMasters" : {
    "type" : "int",
    "defaultValue" : 3,
    "minValue" : 2,
    "maxValue" : 30,
    "metadata" : {
      "description" : "Number of OpenShift masters to deploy"
    }
  },
  "sshKeyData" : {
    "type" : "securestring",
    "metadata" : {
      "description" : "SSH RSA public key file as a string"
    }
  },
  "privateDNSZoneName" : {
    "type" : "string",
    "metadata" : {
      "description" : "Name of the private DNS zone the master nodes are going to be attached to"
    }
  },
  "masterVMSize" : {
    "type" : "string",
    "defaultValue" : "Standard_D8s_v3",
    "allowedValues" : [
      "Standard_A2",
      "Standard_A3",
      "Standard_A4",
      "Standard_A5",
      "Standard_A6",
      "Standard_A7",
      "Standard_A8",
      "Standard_A9",
      "Standard_A10",
      "Standard_A11",
      "Standard_D2",
      "Standard_D3",
      "Standard_D4",
      "Standard_D11",
      "Standard_D12",
      "Standard_D13",
      "Standard_D14",
      "Standard_D2_v2",
      "Standard_D3_v2",
      "Standard_D4_v2",
      "Standard_D5_v2",
      "Standard_D8_v3",
      "Standard_D11_v2",
      "Standard_D12_v2",
      "Standard_D13_v2",
      "Standard_D14_v2",
    ]
  }
}

```

```
"Standard_E2_v3",
"Standard_E4_v3",
"Standard_E8_v3",
"Standard_E16_v3",
"Standard_E32_v3",
"Standard_E64_v3",
"Standard_E2s_v3",
"Standard_E4s_v3",
"Standard_E8s_v3",
"Standard_E16s_v3",
"Standard_E32s_v3",
"Standard_E64s_v3",
"Standard_G1",
"Standard_G2",
"Standard_G3",
"Standard_G4",
"Standard_G5",
"Standard_DS2",
"Standard_DS3",
"Standard_DS4",
"Standard_DS11",
"Standard_DS12",
"Standard_DS13",
"Standard_DS14",
"Standard_DS2_v2",
"Standard_DS3_v2",
"Standard_DS4_v2",
"Standard_DS5_v2",
"Standard_DS11_v2",
"Standard_DS12_v2",
"Standard_DS13_v2",
"Standard_DS14_v2",
"Standard_GS1",
"Standard_GS2",
"Standard_GS3",
"Standard_GS4",
"Standard_GS5",
"Standard_D2s_v3",
"Standard_D4s_v3",
"Standard_D8s_v3"
],
"metadata" : {
  "description" : "The size of the Master Virtual Machines"
}
},
"diskSizeGB" : {
  "type" : "int",
  "defaultValue" : 1024,
  "metadata" : {
    "description" : "Size of the Master VM OS disk, in GB"
  }
}
},
"variables" : {
  "location" : "[resourceGroup().location]",
  "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
```



```

"virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
"masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
"masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
"masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
"internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
"sshKeyPath" : "/home/core/.ssh/authorized_keys",
"identityName" : "[concat(parameters('baseName'), '-identity')]",
"imageName" : "[concat(parameters('baseName'), '-image')]",
"copy" : [
  {
    "name" : "vmNames",
    "count" : "[parameters('numberOfMasters')]",
    "input" : "[concat(parameters('baseName'), '-master-', copyIndex('vmNames'))]"
  }
]
},
"resources" : [
  {
    "apiVersion" : "2018-06-01",
    "type" : "Microsoft.Network/networkInterfaces",
    "copy" : {
      "name" : "nicCopy",
      "count" : "[length(variables('vmNames'))]"
    },
    "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
    "location" : "[variables('location')]",
    "properties" : {
      "ipConfigurations" : [
        {
          "name" : "pipConfig",
          "properties" : {
            "privateIPAllocationMethod" : "Dynamic",
            "subnet" : {
              "id" : "[variables('masterSubnetRef')]"
            },
            "loadBalancerBackendAddressPools" : [
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
              },
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
              }
            ]
          }
        }
      ]
    }
  }
]
},
{
  "apiVersion": "2018-09-01",

```

```

"type": "Microsoft.Network/privateDnsZones/SRV",
"name": "[concat(parameters('privateDNSZoneName'), '/_etcd-server-ssl._tcp')]",
"location" : "[variables('location')]",
"properties": {
  "ttl": 60,
  "copy": [{
    "name": "srvRecords",
    "count": "[length(variables('vmNames'))]",
    "input": {
      "priority": 0,
      "weight" : 10,
      "port" : 2380,
      "target" : "[concat('etcd-', copyIndex('srvRecords'), '.',
parameters('privateDNSZoneName'))]"
    }
  ]
}
},
{
  "apiVersion": "2018-09-01",
  "type": "Microsoft.Network/privateDnsZones/A",
  "copy" : {
    "name" : "dnsCopy",
    "count" : "[length(variables('vmNames'))]"
  },
  "name": "[concat(parameters('privateDNSZoneName'), '/etcd-', copyIndex())]",
  "location" : "[variables('location')]",
  "dependsOn" : [
    "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]"
  ],
  "properties": {
    "ttl": 60,
    "aRecords": [
      {
        "ipv4Address": "[reference(concat(variables('vmNames')[copyIndex()], '-
nic')).ipConfigurations[0].properties.privateIPAddress]"
      }
    ]
  }
},
{
  "apiVersion" : "2018-06-01",
  "type" : "Microsoft.Compute/virtualMachines",
  "copy" : {
    "name" : "vmCopy",
    "count" : "[length(variables('vmNames'))]"
  },
  "name" : "[variables('vmNames')[copyIndex()]]",
  "location" : "[variables('location')]",
  "identity" : {
    "type" : "userAssigned",
    "userAssignedIdentities" : {
      "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
    }
  }
}

```

```

    },
    "dependsOn" : [
      "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-nic'))]",
      "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'), '/A/etcd-', copyIndex())]",
      "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'), '/SRV/_etcd-server-ssl._tcp')]"
    ],
    "properties" : {
      "hardwareProfile" : {
        "vmSize" : "[parameters('masterVMSize')]"
      },
      "osProfile" : {
        "computerName" : "[variables('vmNames')[copyIndex()]]",
        "adminUsername" : "core",
        "customData" : "[parameters('masterIgnition')]",
        "linuxConfiguration" : {
          "disablePasswordAuthentication" : true,
          "ssh" : {
            "publicKeys" : [
              {
                "path" : "[variables('sshKeyPath')]",
                "keyData" : "[parameters('sshKeyData')]"
              }
            ]
          }
        }
      },
      "storageProfile" : {
        "imageReference": {
          "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
        },
        "osDisk" : {
          "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
          "osType" : "Linux",
          "createOption" : "FromImage",
          "caching": "ReadOnly",
          "writeAcceleratorEnabled": false,
          "managedDisk": {
            "storageAccountType": "Premium_LRS"
          },
          "diskSizeGB" : "[parameters('diskSizeGB')]"
        }
      },
      "networkProfile" : {
        "networkInterfaces" : [
          {
            "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')[copyIndex()], '-nic'))]",
            "properties": {
              "primary": false
            }
          }
        ]
      }
    }
  }
}

```



3.8.16. Wait for bootstrap completion and remove bootstrap resources in Azure

After you create all of the required infrastructure in Microsoft Azure, wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.
- Create and configure a VNet and associated subnets in Azure.
- Create and configure networking and load balancers in Azure.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level info 2
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

2. Delete the bootstrap resources:

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
```

```
{CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-ssh-pip
```

3.8.17. Creating additional worker machines in Azure

You can create worker machines in Microsoft Azure for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Azure Resource Manager (ARM) template. Additional instances can be launched by including additional resources of type **06_workers.json** in the file.



NOTE

If you do not use the provided ARM template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure an Azure account.
- Generate the Ignition config files for your cluster.
- Create and configure a VNet and associated subnets in Azure.
- Create and configure networking and load balancers in Azure.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Copy the template from the **ARM template for worker machines** section of this topic and save it as **06_workers.json** in your cluster's installation directory. This template describes the worker machines that your cluster requires.
2. Export the following variable needed by the worker machine deployment:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64`
```

3. Create the deployment by using the **az** CLI:

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
```

```
--parameters sshKeyData="${SSH_KEY}" \ 2
--parameters baseName="${INFRA_ID}" 3
```

- 1** The Ignition content for the worker nodes.
- 2** The SSH RSA public key file as a string.
- 3** The base name to be used in resource names; this is usually the cluster's infrastructure ID.

3.8.17.1. ARM template for worker machines

You can use the following Azure Resource Manager (ARM) template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 3.6. 06_workers.json ARM template

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "workerIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the worker nodes"
      }
    },
    "numberOfNodes" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift compute nodes to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string"
      }
    },
    "nodeVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "allowedValues" : [
        "Standard_A2",
```

"Standard_A3",
"Standard_A4",
"Standard_A5",
"Standard_A6",
"Standard_A7",
"Standard_A8",
"Standard_A9",
"Standard_A10",
"Standard_A11",
"Standard_D2",
"Standard_D3",
"Standard_D4",
"Standard_D11",
"Standard_D12",
"Standard_D13",
"Standard_D14",
"Standard_D2_v2",
"Standard_D3_v2",
"Standard_D4_v2",
"Standard_D5_v2",
"Standard_D8_v3",
"Standard_D11_v2",
"Standard_D12_v2",
"Standard_D13_v2",
"Standard_D14_v2",
"Standard_E2_v3",
"Standard_E4_v3",
"Standard_E8_v3",
"Standard_E16_v3",
"Standard_E32_v3",
"Standard_E64_v3",
"Standard_E2s_v3",
"Standard_E4s_v3",
"Standard_E8s_v3",
"Standard_E16s_v3",
"Standard_E32s_v3",
"Standard_E64s_v3",
"Standard_G1",
"Standard_G2",
"Standard_G3",
"Standard_G4",
"Standard_G5",
"Standard_DS2",
"Standard_DS3",
"Standard_DS4",
"Standard_DS11",
"Standard_DS12",
"Standard_DS13",
"Standard_DS14",
"Standard_DS2_v2",
"Standard_DS3_v2",
"Standard_DS4_v2",
"Standard_DS5_v2",
"Standard_DS11_v2",
"Standard_DS12_v2",
"Standard_DS13_v2",

```

    "Standard_DS14_v2",
    "Standard_GS1",
    "Standard_GS2",
    "Standard_GS3",
    "Standard_GS4",
    "Standard_GS5",
    "Standard_D2s_v3",
    "Standard_D4s_v3",
    "Standard_D8s_v3"
  ],
  "metadata" : {
    "description" : "The size of the each Node Virtual Machine"
  }
},
"variables" : {
  "location" : "[resourceGroup().location]",
  "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
  "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
  "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
  "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('nodeSubnetName'))]",
  "infraLoadBalancerName" : "[parameters('baseName')]",
  "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
  "identityName" : "[concat(parameters('baseName'), '-identity')]",
  "imageName" : "[concat(parameters('baseName'), '-image')]",
  "copy" : [
    {
      "name" : "vmNames",
      "count" : "[parameters('numberOfNodes')]",
      "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-',
copyIndex('vmNames', 1))]"
    }
  ]
},
"resources" : [
  {
    "apiVersion" : "2019-05-01",
    "name" : "[concat('node', copyIndex())]",
    "type" : "Microsoft.Resources/deployments",
    "copy" : {
      "name" : "nodeCopy",
      "count" : "[length(variables('vmNames'))]"
    },
    "properties" : {
      "mode" : "Incremental",
      "template" : {
        "$schema" : "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
        "contentVersion" : "1.0.0.0",
        "resources" : [
          {
            "apiVersion" : "2018-06-01",
            "type" : "Microsoft.Network/networkInterfaces",
            "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",

```



```

"location" : "[variables('location')]",
"properties" : {
  "ipConfigurations" : [
    {
      "name" : "pipConfig",
      "properties" : {
        "privateIPAllocationMethod" : "Dynamic",
        "subnet" : {
          "id" : "[variables('nodeSubnetRef')]"
        }
      }
    }
  ]
},
{
  "apiVersion" : "2018-06-01",
  "type" : "Microsoft.Compute/virtualMachines",
  "name" : "[variables('vmNames')[copyIndex()]]",
  "location" : "[variables('location')]",
  "tags" : {
    "kubernetes.io-cluster-ffranzupi": "owned"
  },
  "identity" : {
    "type" : "userAssigned",
    "userAssignedIdentities" : {
      "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
    }
  },
  "dependsOn" : [
    "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
[copyIndex()], '-nic'))]"
  ],
  "properties" : {
    "hardwareProfile" : {
      "vmSize" : "[parameters('nodeVMSize')]"
    },
    "osProfile" : {
      "computerName" : "[variables('vmNames')[copyIndex()]]",
      "adminUsername" : "capi",
      "customData" : "[parameters('workerIgnition')]",
      "linuxConfiguration" : {
        "disablePasswordAuthentication" : true,
        "ssh" : {
          "publicKeys" : [
            {
              "path" : "[variables('sshKeyPath')]",
              "keyData" : "[parameters('sshKeyData')]"
            }
          ]
        }
      }
    }
  },
  "storageProfile" : {
    "imageReference": {

```

```

        "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
      },
      "osDisk" : {
        "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
        "osType" : "Linux",
        "createOption" : "FromImage",
        "managedDisk": {
          "storageAccountType": "Premium_LRS"
        },
        "diskSizeGB": 128
      }
    },
    "networkProfile" : {
      "networkInterfaces" : [
        {
          "id" : "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('vmNames')[copyIndex()], '-nic'))]",
          "properties": {
            "primary": true
          }
        }
      ]
    }
  }
}

```

3.8.18. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

3.8.18.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.

3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.8.18.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**. To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

3.8.18.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

3.8.19. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

3.8.20. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready     master   63m   v1.18.3
master-1  Ready     master   63m   v1.18.3
master-2  Ready     master   64m   v1.18.3
worker-0  NotReady  worker   76s   v1.18.3
worker-1  NotReady  worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   73m   v1.20.0
master-1  Ready   master   73m   v1.20.0
master-2  Ready   master   74m   v1.20.0
worker-0  Ready   worker   11m   v1.20.0
worker-1  Ready   worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

3.8.21. Adding the Ingress DNS records

If you removed the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs, you must manually create DNS records that point at the Ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}**, or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- You deployed an OpenShift Container Platform cluster on Microsoft Azure by using infrastructure that you provisioned.
- Install the OpenShift CLI (**oc**).
- Install the **jq** package.
- Install or update the [Azure CLI](#).

Procedure

1. Confirm the Ingress router has created a load balancer and populated the **EXTERNAL-IP** field:

```
$ oc -n openshift-ingress get service router-default
```

Example output

```
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
router-default LoadBalancer  172.30.20.10  35.130.120.110 80:32288/TCP,443:31215/TCP 20
```

2. Export the Ingress router IP as a variable:

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
```

3. Add a ***.apps** record to the public DNS zone.

- a. If you are adding this cluster to a new public zone, run:

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

- b. If you are adding this cluster to an already existing public zone, run:

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

4. Add a ***.apps** record to the private DNS zone:

- a. Create a ***.apps** record by using the following command:

```
$ az network private-dns record-set a create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps --ttl 300
```

- b. Add the ***.apps** record to the private DNS zone by using the following command:

```
$ az network private-dns record-set a add-record -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER}
```

If you prefer to add explicit domains instead of using a wildcard, you can create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

Example output

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
grafana-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

3.8.22. Completing an Azure installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Microsoft Azure user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned Azure infrastructure.
- Install the **oc** CLI and log in.

Procedure

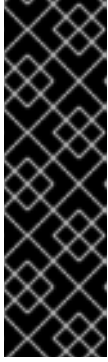
- Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

3.9. UNINSTALLING A CLUSTER ON AZURE

You can remove a cluster that you deployed to Microsoft Azure.

3.9.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access.

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```

$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2

```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

CHAPTER 4. INSTALLING ON GCP

4.1. CONFIGURING A GCP PROJECT

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

4.1.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>.<base_domain>** URL; the Premium Tier is required for internal load balancing.

4.1.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See [Enabling services](#) in the GCP documentation.

Table 4.1. Required API services

API service	Console service name
Compute Engine API	compute.googleapis.com
Google Cloud APIs	cloudapis.googleapis.com
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com

API service	Console service name
IAM Service Account Credentials API	iamcredentials.googleapis.com
Identity and Access Management (IAM) API	iam.googleapis.com
Service Management API	servicemanagement.googleapis.com
Service Usage API	serviceusage.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

4.1.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.



NOTE

If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see [Google Domains](#).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](#) in the GCP documentation. Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
3. Extract the new authoritative name servers from the hosted zone records. See [Look up your Cloud DNS name servers](#) in the GCP documentation. You typically have four name servers.
4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: [How to switch to custom name servers](#).
5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.

6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

4.1.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default [Quotas](#) do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 4.2. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Compute	Global	11	1
Forwarding rules	Compute	Global	2	0
In-use global IP addresses	Compute	Global	4	1
Health checks	Compute	Global	3	0
Images	Compute	Global	1	0
Networks	Compute	Global	2	0
Static IP addresses	Compute	Region	4	1
Routers	Compute	Global	1	0
Routes	Compute	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Compute	Global	3	0
CPUs	Compute	Region	28	4
Persistent disk SSD (GB)	Compute	Region	896	128

**NOTE**

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europa-north1**
- **europa-west2**
- **europa-west3**
- **europa-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

4.1.5. Creating a service account in GCP

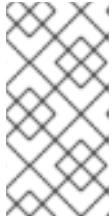
OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See [Creating a service account](#) in the GCP documentation.
2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).

**NOTE**

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See [Creating service account keys](#) in the GCP documentation.
The service account key is required to create a cluster.

4.1.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Security Admin
- Service Account Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 4.3. GCP service account permissions

Account	Roles
Control Plane	roles/compute.instanceAdmin
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin

Account	Roles
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin

4.1.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europa-north1** (Hamina, Finland)
- **europa-west1** (St. Ghislain, Belgium)
- **europa-west2** (London, England, UK)
- **europa-west3** (Frankfurt, Germany)
- **europa-west4** (Eemshaven, Netherlands)
- **europa-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.1.7. Next steps

- Install an OpenShift Container Platform cluster on GCP. You can [install a customized cluster](#) or [quickly install a cluster](#) with default options.

4.2. MANUALLY CREATING IAM FOR GCP

4.2.1. Manually create IAM

The Cloud Credential Operator can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

Procedure

1. Run the OpenShift Container Platform installer to generate manifests:

```
$ openshift-install create manifests --dir=mycluster
```

2. Insert a config map into the manifests directory so that the Cloud Credential Operator is placed in manual mode:

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-credential-operator-config
  namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF
```

3. Remove the **admin** credential secret created using your local cloud credentials. This removal prevents your **admin** credential from being stored in the cluster:

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. Obtain the OpenShift Container Platform release image your **openshift-install** binary is built to use:

```
$ bin/openshift-install version
```

Example output

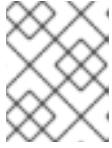
```
release image quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64
```

5. Locate all **CredentialsRequest** objects in this release image that target the cloud you are deploying on:

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64 --to
./release-image
```


6. Locate the **CredentialsRequests** in the extracted file:

```
$ grep -l "apiVersion: cloudcredential.openshift.io" * | xargs cat
```



NOTE

In a future OpenShift Container Platform release, there will be a new **oc adm release** command to scan for the **CredentialsRequests** and display them.

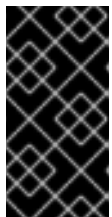
This displays the details for each request. Remember to ignore any **CredentialsRequests** where the **spec.providerSpec.kind** does not match the cloud provider you are installing to.

Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-gcs
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
      - roles/storage.admin
      - roles/iam.serviceAccountUser
    skipServiceCheck: true
```

7. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in each **request.spec.secretRef**. The format for the secret data varies for each cloud provider.
8. Proceed with cluster creation:

```
$ openshift-install create cluster --dir=mycluster
```



IMPORTANT

Before performing an upgrade, you might need to adjust your credentials if permissions have changed in the next release. In the future, the Cloud Credential Operator might prevent you from upgrading until you have indicated that you have addressed updated permissions.

4.2.2. Admin credentials root secret format

Each cloud provider uses a credentials root secret in the **kube-system** namespace by convention, which is then used to satisfy all credentials requests and create their respective secrets. This is done either by minting new credentials, *Mint Mode*, or by copying the credentials root secret, *Passthrough Mode*.

The format for the secret varies by cloud, and is also used for each **CredentialsRequest** secret.

Google Cloud Platform (GCP) secret format

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <ServiceAccount>
```

4.2.2.1. Upgrades

In a future release, improvements to the Cloud Credential Operator will prevent situations where a user might enter an upgrade that will fail because their manually maintained credentials have not been updated to match the **CredentialsRequest** objects in the upcoming release image.

4.2.3. Mint Mode

Mint Mode is supported for AWS, GCP, and Azure.

The default and recommended best practice for running OpenShift Container Platform is to run the installer with an administrator-level cloud credential. The **admin** credential is stored in the **kube-system** namespace, and then used by the Cloud Credential Operator to process the **CredentialsRequest** objects in the cluster and create new users for each with specific permissions.

The benefits of Mint Mode include:

- Each cluster component only has the permissions it requires.
- Automatic, on-going reconciliation for cloud credentials including upgrades, which might require additional credentials or permissions.

One drawback is that Mint Mode requires **admin** credential storage in a cluster **kube-system** secret.

4.3. INSTALLING A CLUSTER QUICKLY ON GCP

In OpenShift Container Platform version 4.5, you can install a cluster on Google Cloud Platform (GCP) that uses the default configuration options.

4.3.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

4.3.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

4.3.3. Generating an SSH private key and adding it to the agent

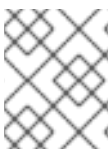
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`
2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.3.4. Obtaining the installation program

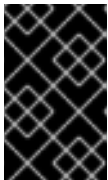
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

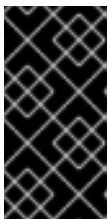
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

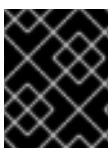
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

4.3.5. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

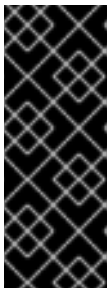
- The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
- The `~/.gcp/osServiceAccount.json` file
- The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

- a. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- b. Select **gcp** as the platform to target.
- c. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- d. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

- e. Select the region to deploy the cluster to.
- f. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- g. Enter a descriptive name for your cluster. If you provide a name that is longer than 6 characters, only the first 6 characters will be used in the infrastructure ID that is generated from the cluster name.
- h. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

4.3.6. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.3.6.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.3.6.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.3.6.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.3.7. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.3.8. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

4.4. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a customized cluster on infrastructure that the installation program provisions on Google Cloud Platform (GCP). To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

4.4.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

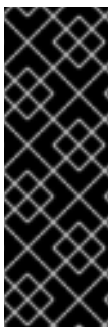
4.4.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

4.4.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

■

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1

Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.4.4. Obtaining the installation program

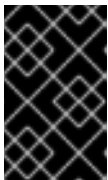
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.

**IMPORTANT**

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

**IMPORTANT**

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

4.4.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$. /openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

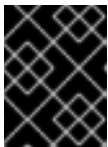


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.

- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

4.4.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

4.4.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 4.4. Required parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object

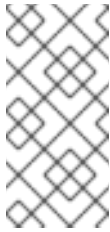
Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


4.4.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 4.5. Network parameters


Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


4.4.5.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 4.6. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

4.4.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

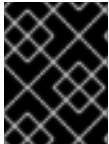
Additional GCP configuration parameters are described in the following table:

Table 4.7. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

4.4.5.2. Sample customized `install-config.yaml` file for GCP

You can customize the `install-config.yaml` file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
metadata:
  name: test-cluster 8
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
pullSecret: '{"auths": ...}' 11
fips: false 12
sshKey: ssh-ed25519 AAAA... 13

```

1 8 9 10 11 Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both

sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

- 4 7** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.

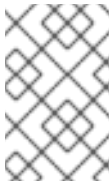


IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

- 12** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.

- 13** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

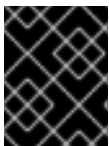


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

4.4.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables

- The `~/gcp/osServiceAccount.json` file
- The `gcloud cli` default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

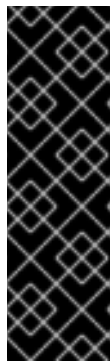
- 1 For `<installation_directory>`, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

4.4.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.4.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.4.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**. To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.4.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.4.8. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.4.9. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

4.5. INSTALLING A CLUSTER ON GCP WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster with a customized network configuration on infrastructure that the installation program provisions on Google Cloud Platform (GCP). By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

4.5.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

4.5.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.

- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

4.5.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.5.4. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

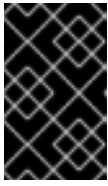
Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

4.5.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

**IMPORTANT**

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

4.5.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for

the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

4.5.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 4.8. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .


Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

4.5.5.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 4.9. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	<p>Object</p>  <p>NOTE</p> <p>You cannot modify parameters specified by the networking object after installation.</p>

Parameter	Description	Values
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.

4.5.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 4.10. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

4.5.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 4.11. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.

Parameter	Description	Values
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.



IMPORTANT

The Open Virtual Networking (OVN) Kubernetes network plug-in is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of the OVN Technology Preview, see <https://access.redhat.com/articles/4380121>.

4.5.5.2. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 4.12. Required network parameters

Parameter	Description	Value
networking.networkType	The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5. The OVNKubernetes plug-in is available as a Technology Preview in OpenShift Container Platform 4.5.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .

Parameter	Description	Value
networking.clusterNetwork[].cidr	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .
networking.clusterNetwork[].hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork[]	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.machineNetwork[].cidr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

4.5.5.3. Sample customized `install-config.yaml` file for GCP

You can customize the `install-config.yaml` file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your `install-config.yaml` file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
compute: 5 6
  - hyperthreading: Enabled 7

```



```

name: worker
platform:
  gcp:
    type: n2-standard-4
    zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
metadata:
  name: test-cluster 8
networking: 9
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 10
    region: us-central1 11
pullSecret: '{"auths": ...}' 12
fips: false 13
sshKey: ssh-ed25519 AAAA... 14

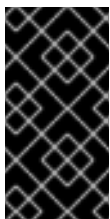
```

1 8 10 11 12 Required. The installation program prompts you for this value.

2 5 9 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



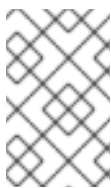
IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography

modules that are provided with RHCOS instead.

- 14 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

4.5.6. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster. Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of **kube-proxy** settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.

Procedure

1. Use the following command to create manifests:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1 For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, several network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

- Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: ❶
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

- The parameters for the **spec** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

- Save the **cluster-network-03-config.yml** file and quit the text editor.
- Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

4.5.7. Cluster Network Operator configuration

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: ❶
```

```

- cidr: 10.128.0.0/14
  hostPrefix: 23
  serviceNetwork: 2
- 172.30.0.0/16
  defaultNetwork: 3
  ...
  kubeProxyConfig: 4
  iptablesSyncPeriod: 30s 5
  proxyArguments:
    iptables-min-sync-period: 6
    - 0s

```

- 1 2 Specified in the **install-config.yaml** file.
- 3 Configures the default Container Network Interface (CNI) network provider for the cluster network.
- 4 The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.
- 5 The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.



NOTE

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

- 6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#).

4.5.7.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```

defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5

```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.

- 4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**.

- 5 The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for VXLAN, since both SDNs use the same default VXLAN port number.

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

4.5.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider

The following YAML object describes the configuration parameters for the OVN-Kubernetes default CNI network provider.

```
defaultNetwork:
  type: OVNKubernetes 1
  ovnKubernetesConfig: 2
    mtu: 1400 3
    genevePort: 6081 4
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OVN-Kubernetes configuration.
- 3 The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **100** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1400**.

- 4 The UDP port for the Geneve overlay network.

4.5.7.3. Cluster Network Operator example configuration

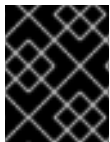
A complete CR object for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 0s
```

4.5.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

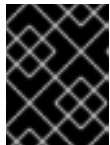
**NOTE**

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

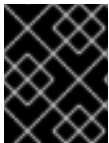
The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

4.5.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.5.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.5.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.5.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:


```
$ oc <command>
```

4.5.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

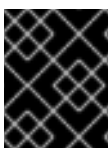
```
system:admin
```

4.5.11. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

4.6. INSTALLING A CLUSTER ON GCP IN A RESTRICTED NETWORK

In OpenShift Container Platform 4.5, you can install a cluster on Google Cloud Platform (GCP) in a restricted network by creating an internal mirror of the installation release content on an existing Google Virtual Private Cloud (VPC).



IMPORTANT

You can install an OpenShift Container Platform cluster by using mirrored installation release content, but your cluster will require internet access to use the GCP APIs.

4.6.1. Prerequisites

- You [mirrored the images for a disconnected installation](#) to your registry and obtained the `imageContentSources` data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- You have an existing VPC in GCP. While installing a cluster in a restricted network that uses installer-provisioned infrastructure, you cannot use the installer-provisioned VPC. You must use a user-provisioned VPC that satisfies one of the following requirements:
 - Contains the mirror registry
 - Has firewall rules or a peering connection to access the mirror registry hosted elsewhere
- You reviewed details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to. While you might need to grant access to more sites, you must grant access to ***.googleapis.com** and **accounts.google.com**.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

4.6.2. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.

4.6.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The `ClusterVersion` status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

4.6.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

4.6.4. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

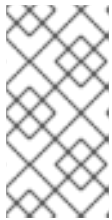
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.6.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your bastion host.
- Have the **imageContentSources** values that were generated during mirror registry creation.
- Obtain the contents of the certificate for your mirror registry.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.

- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.
 - a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email": "you@example.com"}}}'
```

For **<bastion_host_name>**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

- b. Add the **additionalTrustBundle** parameter and value.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  /-----END CERTIFICATE-----
```

The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

- c. Define the network and subnets for the VPC to install the cluster in under the parent **platform.gcp** field:

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

For **platform.gcp.network**, specify the name for the existing Google VPC. For **platform.gcp.controlPlaneSubnet** and **platform.gcp.computeSubnet**, specify the existing subnets to deploy the control plane machines and compute machines, respectively.

- d. Add the image content resources, which look like this excerpt:

```
imageContentSources:
- mirrors:
```

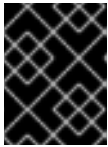
```

- <bastion_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
- <bastion_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release

```

To complete these values, use the **imageContentSources** that you recorded during mirror registry creation.

3. Make any other modifications to the **install-config.yaml** file that you require. You can find more information about the available parameters in the **Installation configuration parameters** section.
4. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

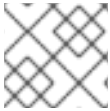


IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

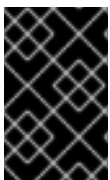
4.6.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

4.6.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 4.13. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object


Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


4.6.5.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 4.14. Network parameters


Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


4.6.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 4.15. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

4.6.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 4.16. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

4.6.5.2. Sample customized `install-config.yaml` file for GCP

You can customize the `install-config.yaml` file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
metadata:
  name: test-cluster 8
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
    network: existing_vpc 11
    controlPlaneSubnet: control_plane_subnet 12
    computeSubnet: compute_subnet 13
pullSecret: '{"auths":{"<local_registry>":{"auth":"<credentials>","email":"you@example.com"}}}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
additionalTrustBundle: | 17
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
imageContentSources: 18

```

- mirrors:
 - <local_registry>/<local_repository_name>/release
 - source: quay.io/openshift-release-dev/ocp-release
- mirrors:
 - <local_registry>/<local_repository_name>/release
 - source: registry.svc.ci.openshift.org/ocp/release

- 1 8 9 10** Required. The installation program prompts you for this value.
- 2 5** If you do not provide these parameters and values, the installation program provides the default value.
- 3 6** The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 4 7** Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

- 11** Specify the name of an existing VPC.
- 12** Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.
- 13** Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.
- 14** For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 15** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 16** You can optionally provide the **sshKey** value that you use to access the machines in your cluster.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

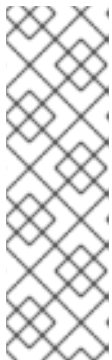
- 17 Provide the contents of the certificate file that you used for your mirror registry.
- 18 Provide the **imageContentSources** section from the output of the command to mirror the repository.

4.6.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

**NOTE**

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```

apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

4.6.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

- The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
- The `~/.gcp/osServiceAccount.json` file
- The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1** For `<installation_directory>`, specify the location of your customized `./install-config.yaml` file.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

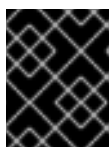
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.

- If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
- If you included the **Service Account Key Admin** role, you can remove it.

4.6.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.6.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.6.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.

4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.6.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.6.8. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

- Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.6.9. Next steps

- [Customize your cluster](#).
- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to [use Operator Lifecycle Manager \(OLM\) on restricted networks](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).

4.7. INSTALLING A CLUSTER ON GCP INTO AN EXISTING VPC

In OpenShift Container Platform version 4.5, you can install a cluster into an existing Virtual Private Cloud (VPC) on Google Cloud Platform (GCP). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

4.7.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

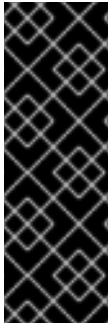
4.7.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, use [subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

4.7.3. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

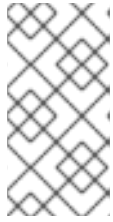
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.7.4. Obtaining the installation program

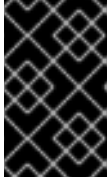
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

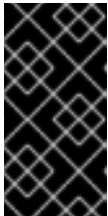
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

4.7.5. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

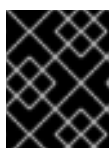
- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

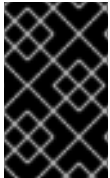
4.7.5.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

4.7.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 4.17. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object

Parameter	Description	Values
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}.{{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


4.7.5.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 4.18. Network parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>


4.7.5.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 4.19. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hypert hreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platfor m	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replica s	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <p> NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <p> NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

4.7.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 4.20. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

4.7.5.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
    replicas: 3
compute: 5 6
  - hyperthreading: Enabled 7
  name: worker
  platform:

```

```

gcp:
  type: n2-standard-4
  zones:
  - us-central1-a
  - us-central1-c
  replicas: 3
metadata:
  name: test-cluster 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
    network: existing_vpc 11
    controlPlaneSubnet: control_plane_subnet 12
    computeSubnet: compute_subnet 13
  pullSecret: '{"auths": ...}' 14
  fips: false 15
  sshKey: ssh-ed25519 AAAA... 16

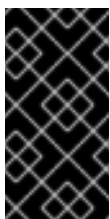
```

1 8 9 10 14 Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

11 Specify the name of an existing VPC.

- 12 Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.
- 13 Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.
- 15 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 16 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

4.7.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

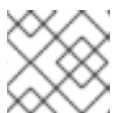
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
```

```

noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...

```

- 1** A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2** A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3** A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4** If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

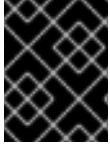


NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

4.7.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
 - The `~/.gcp/osServiceAccount.json` file
 - The **gcloud cli** default credentials
2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For `<installation_directory>`, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



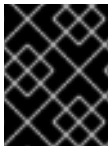
IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

4.7.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.7.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.7.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.7.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.7.8. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.7.9. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

4.8. INSTALLING A PRIVATE CLUSTER ON GCP

In OpenShift Container Platform version 4.5, you can install a private cluster into an existing VPC on Google Cloud Platform (GCP). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

4.8.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- [Configure a GCP account](#) to host the cluster.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

4.8.2. Private clusters

If your environment does not require an external Internet connection, you can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the Internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

To deploy a private cluster, you must use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.

Additionally, you must deploy a private cluster from a machine that has access the API services for the cloud you provision to, the hosts on the network that you provision, and to the internet to obtain installation media. You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

4.8.2.1. Private clusters in GCP

To create a private cluster on Google Cloud Platform (GCP), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for only internal traffic.

The cluster still requires access to Internet to access the GCP APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public network load balancers, which support public ingress
- A public DNS zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private DNS zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

Because it is not possible to limit access to external load balancers based on source tags, the private cluster uses only internal load balancers to allow access to internal instances.

The internal load balancer relies on instance groups rather than the target pools that the network load balancers use. The installation program creates instance groups for each zone, even if there is no instance in that group.

- The cluster IP address is internal only.
- One forwarding rule manages both the Kubernetes API and machine config server ports.
- The backend service is comprised of each zone's instance group and, while it exists, the bootstrap instance group.
- The firewall uses a single rule that is based on only internal source ranges.

4.8.2.1.1. Limitations

No health check for the Machine config server, **/healthz**, runs because of a difference in load balancer functionality. Two internal load balancers cannot share a single IP address, but two network load balancers can share a single external IP address. Instead, the health of an instance is determined entirely by the **/readyz** check on port 6443.

4.8.3. About using a custom VPC

In OpenShift Container Platform 4.5, you can deploy a cluster into an existing VPC in Google Cloud Platform (GCP). If you do, you must also use existing subnets within the VPC and routing rules.

By deploying OpenShift Container Platform into an existing GCP VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. This is a good option to use if you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself.

4.8.3.1. Requirements for using your VPC

The installation program will no longer create the following components:

- VPC
- Subnets
- Cloud router
- Cloud NAT
- NAT IP addresses

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. The installation program cannot subdivide network ranges for the cluster to use, set route tables for the subnets, or set VPC options like DHCP, so you must do so before you install the cluster.

Your VPC and subnets must meet the following characteristics:

- The VPC must be in the same GCP project that you deploy the OpenShift Container Platform cluster to.
- To allow access to the Internet from the control plane and compute machines, you must configure cloud NAT on the subnets to allow egress to it. These machines do not have a public address. Even if you do not require access to the Internet, you must allow egress to the VPC network to obtain the installation program and images. Because multiple cloud NATs cannot be configured on the shared subnets, the installation program cannot configure it.

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist and belong to the VPC that you specified.
- The subnet CIDRs belong to the machine CIDR.
- You must provide a subnet to deploy the cluster control plane and compute machines to. You can use the same subnet for both machine types.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted.

4.8.3.2. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different

resources in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or Ingress rules.

The GCP credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as load balancers, security groups, storage, and nodes.

4.8.3.3. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is preserved by firewall rules that reference the machines in your cluster by the cluster's infrastructure ID. Only traffic within the cluster is allowed.

If you deploy multiple clusters to the same VPC, the following components might share access between clusters:

- The API, which is globally available with an external publishing strategy or available throughout the network in an internal publishing strategy
- Debugging tools, such as ports on VM instances that are open to the machine CIDR for SSH and ICMP access

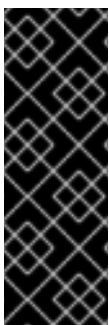
4.8.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

4.8.5. Generating an SSH private key and adding it to the agent

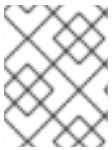
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`
- Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

- Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

4.8.6. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

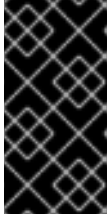
Procedure

- Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
- Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

4.8.7. Manually creating the installation configuration file

For installations of a private OpenShift Container Platform cluster that are only accessible from an internal network and are not visible to the Internet, you must manually generate your installation configuration file.

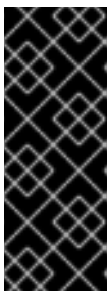
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

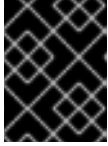
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

4.8.7.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

4.8.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 4.21. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .

Parameter	Description	Values
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


4.8.7.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 4.22. Network parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p>  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>


4.8.7.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 4.23. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

4.8.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 4.24. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.

Parameter	Description	Values
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

4.8.7.2. Sample customized `install-config.yaml` file for GCP

You can customize the `install-config.yaml` file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your `install-config.yaml` file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2 3
  hyperthreading: Enabled 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
compute: 5 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
metadata:
  name: test-cluster 8
networking:

```

```

clusterNetwork:
- cidr: 10.128.0.0/14
  hostPrefix: 23
machineNetwork:
- cidr: 10.0.0.0/16
networkType: OpenShiftSDN
serviceNetwork:
- 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
    network: existing_vpc 11
    controlPlaneSubnet: control_plane_subnet 12
    computeSubnet: compute_subnet 13
  pullSecret: '{"auths": ...}' 14
  fips: false 15
  sshKey: ssh-ed25519 AAAA... 16
  publish: Internal 17

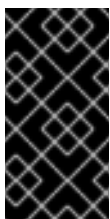
```

1 8 9 10 14 Required. The installation program prompts you for this value.

2 5 If you do not provide these parameters and values, the installation program provides the default value.

3 6 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

4 7 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

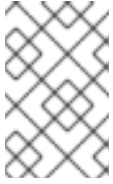
11 Specify the name of an existing VPC.

12 Specify the name of the existing subnet to deploy the control plane machines to. The subnet must belong to the VPC that you specified.

13 Specify the name of the existing subnet to deploy the compute machines to. The subnet must belong to the VPC that you specified.

15 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is

- 16 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 17 How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the Internet. The default value is **External**.

4.8.7.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

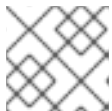
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

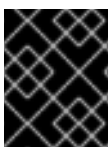


NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

4.8.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

4.8.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.8.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.8.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.8.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.

3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.8.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

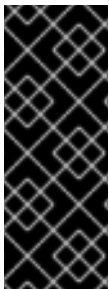
4.8.11. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

4.9. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.5, you can install a cluster on Google Cloud Platform (GCP) that uses infrastructure that you provide.

The steps for performing a user-provided infrastructure install are outlined here. Several [Deployment Manager](#) templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

4.9.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.



NOTE

Be sure to also review this site list if you are configuring a proxy.

4.9.2. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

4.9.3. Configuring your GCP project

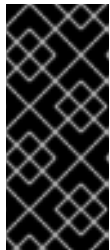
Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

4.9.3.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>.<base_domain>** URL; the Premium Tier is required for internal load balancing.

4.9.3.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See [Enabling services](#) in the GCP documentation.

Table 4.25. Required API services

API service	Console service name
Cloud Deployment Manager V2 API	deploymentmanager.googleapis.com
Compute Engine API	compute.googleapis.com
Google Cloud APIs	cloudapis.googleapis.com
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com
IAM Service Account Credentials API	iamcredentials.googleapis.com
Identity and Access Management (IAM) API	iam.googleapis.com
Service Management API	servicemanagement.googleapis.com

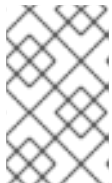
API service	Console service name
Service Usage API	serviceusage.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

4.9.3.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.



NOTE

If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see [Google Domains](#).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](#) in the GCP documentation. Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
3. Extract the new authoritative name servers from the hosted zone records. See [Look up your Cloud DNS name servers](#) in the GCP documentation. You typically have four name servers.
4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: [How to switch to custom name servers](#).
5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

4.9.3.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default [Quotas](#) do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 4.26. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Networking	Global	11	1
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0



NOTE

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europa-north1**

- **europa-west2**
- **europa-west3**
- **europa-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

4.9.3.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See [Creating a service account](#) in the GCP documentation.
2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See [Creating service account keys](#) in the GCP documentation.
The service account key is required to create a cluster.

4.9.3.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Security Admin
- Service Account Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor
- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 4.27. GCP service account permissions

Account	Roles
Control Plane	roles/compute.instanceAdmin
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin

4.9.3.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europa-north1** (Hamina, Finland)
- **europa-west1** (St. Ghislain, Belgium)
- **europa-west2** (London, England, UK)
- **europa-west3** (Frankfurt, Germany)
- **europa-west4** (Eemshaven, Netherlands)
- **europa-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.9.3.7. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.
- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **\$PATH**:
 - **gcloud**

- **gsutil**

See [Install the latest Cloud SDK version](#) in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.
See [Authorizing with a service account](#) in the GCP documentation.

4.9.4. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files.

4.9.4.1. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
- c. Optional: If you do not want the cluster to provision compute machines, empty the compute pool by editing the resulting **install-config.yaml** file to set **replicas** to **0** for the **compute** pool:

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 Set to **0**.

2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

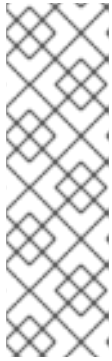
The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

4.9.4.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

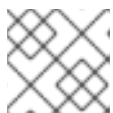
Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you

use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.



NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

4.9.4.3. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```


- 1 For `<installation_directory>`, specify the installation directory that contains the `install-config.yaml` file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:

- a. Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
- b. Locate the `mastersSchedulable` parameter and set its value to `False`.
- c. Save and exit the file.

5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- 1 2 Remove this section completely.

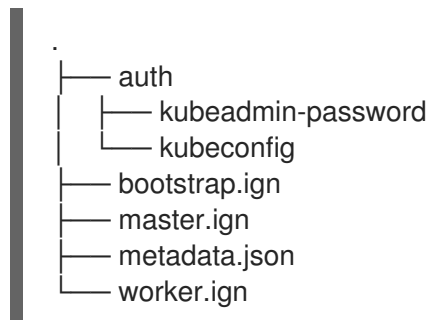
If you do so, you must add ingress DNS records manually in a later step.

6. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:



Additional resources

- [Optional: Adding the ingress DNS records](#)

4.9.5. Exporting common variables

4.9.5.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

4.9.5.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).



NOTE

Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

1. Export the following common variables to be used by the provided Deployment Manager templates:

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

4.9.6. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC** section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.
2. Create a **01_vpc.yaml** resource definition file:

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' ❶
    region: '${REGION}' ❷
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' ❸
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' ❹
EOF
```

- ❶ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- ❷ **region** is the region to deploy the cluster into, for example **us-central1**.
- ❸ **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.
- ❹ **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

4.9.6.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

Example 4.1. 01_vpc.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
```

```

    'autoCreateSubnetworks': False
  }
}, {
  'name': context.properties['infra_id'] + '-master-subnet',
  'type': 'compute.v1.subnetwork',
  'properties': {
    'region': context.properties['region'],
    'network': '${ref.' + context.properties['infra_id'] + '-network.selfLink}',
    'ipCidrRange': context.properties['master_subnet_cidr']
  }
}, {
  'name': context.properties['infra_id'] + '-worker-subnet',
  'type': 'compute.v1.subnetwork',
  'properties': {
    'region': context.properties['region'],
    'network': '${ref.' + context.properties['infra_id'] + '-network.selfLink}',
    'ipCidrRange': context.properties['worker_subnet_cidr']
  }
}, {
  'name': context.properties['infra_id'] + '-router',
  'type': 'compute.v1.router',
  'properties': {
    'region': context.properties['region'],
    'network': '${ref.' + context.properties['infra_id'] + '-network.selfLink}',
    'nats': [{
      'name': context.properties['infra_id'] + '-nat-master',
      'natIpAllocateOption': 'AUTO_ONLY',
      'minPortsPerVm': 7168,
      'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
      'subnetworks': [{
        'name': '${ref.' + context.properties['infra_id'] + '-master-subnet.selfLink}',
        'sourceIpRangesToNat': ['ALL_IP_RANGES']
      }]
    }]
}, {
  'name': context.properties['infra_id'] + '-nat-worker',
  'natIpAllocateOption': 'AUTO_ONLY',
  'minPortsPerVm': 512,
  'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
  'subnetworks': [{
    'name': '${ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink}',
    'sourceIpRangesToNat': ['ALL_IP_RANGES']
  }]
}]
}
]]
return {'resources': resources}

```

4.9.7. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 4.28. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 4.29. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 4.30. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 4.31. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 4.32. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

4.9.8. Creating load balancers in GCP

You must configure load balancers in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.

2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.
3. Export the variables that the deployment template uses:

- a. Export the cluster network location:

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
```

- b. Export the control plane subnet location:

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

- c. Export the three zones that the cluster uses:

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
```

4. Create a **02_infra.yaml** resource definition file:

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py 1
resources:
- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

1 **2** Required only when deploying an external cluster.

- 3 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 4 **region** is the region to deploy the cluster into, for example **us-central1**.
- 5 **control_subnet** is the URI to the control subnet.
- 6 **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

4.9.8.1. Deployment Manager template for the external load balancer

You can use the following Deployment Manager template to deploy the external load balancer that you need for your OpenShift Container Platform cluster:

Example 4.2. 02_lb_ext.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$ (ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
```

```

        'instances': []
    }
}, {
    'name': context.properties['infra_id'] + '-api-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
        'region': context.properties['region'],
        'IPAddress': '${ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink}',
        'target': '${ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink}',
        'portRange': '6443'
    }
}
]]

return {'resources': resources}

```

4.9.8.2. Deployment Manager template for the internal load balancer

You can use the following Deployment Manager template to deploy the internal load balancer that you need for your OpenShift Container Platform cluster:

Example 4.3. 02_lb_int.py Deployment Manager template

```

def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '${ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink}'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',

```

```

    'properties': {
      'backends': backends,
      'healthChecks': ['$$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
      'loadBalancingScheme': 'INTERNAL',
      'region': context.properties['region'],
      'protocol': 'TCP',
      'timeoutSec': 120
    }
  }, {
    'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
      'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
      'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
      'loadBalancingScheme': 'INTERNAL',
      'ports': ['6443', '22623'],
      'region': context.properties['region'],
      'subnetwork': context.properties['control_subnet']
    }
  }
}]

for zone in context.properties['zones']:
  resources.append({
    'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
    'type': 'compute.v1.instanceGroup',
    'properties': {
      'namedPorts': [
        {
          'name': 'ignition',
          'port': 22623
        }, {
          'name': 'https',
          'port': 6443
        }
      ],
      'network': context.properties['cluster_network'],
      'zone': zone
    }
  })

return {'resources': resources}

```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

4.9.9. Creating a private DNS zone in GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.
2. Create a **02_dns.yaml** resource definition file:

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.
- 3** **cluster_network** is the **selfLink** URL to the cluster network.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:
 - a. Add the internal DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
```

```

api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone

```

- b. For an external cluster, also add the external DNS entries:

```

$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}

```

4.9.9.1. Deployment Manager template for the private DNS

You can use the following Deployment Manager template to deploy the private DNS that you need for your OpenShift Container Platform cluster:

Example 4.4. `02_dns.py` Deployment Manager template

```

def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}

```

4.9.10. Creating firewall rules in GCP

You must create firewall rules in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.
2. Create a **03_firewall.yaml** resource definition file:

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' 1
    infra_id: '${INFRA_ID}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
    network_cidr: '${NETWORK_CIDR}' 4
EOF
```

- 1 **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **\${NETWORK_CIDR}**.
- 2 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 3 **cluster_network** is the **selfLink** URL to the cluster network.
- 4 **network_cidr** is the CIDR of the VPC network, for example **10.0.0.0/16**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

4.9.10.1. Deployment Manager template for firewall rules

You can use the following Deployment Manager template to deploy the firewall rules that you need for your OpenShift Container Platform cluster:

Example 4.5. 03_firewall.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    ], {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
```



```

'name': context.properties['infra_id'] + '-control-plane',
'type': 'compute.v1.firewall',
'properties': {
  'network': context.properties['cluster_network'],
  'allowed': [{
    'IPProtocol': 'tcp',
    'ports': ['10257']
  },{
    'IPProtocol': 'tcp',
    'ports': ['10259']
  },{
    'IPProtocol': 'tcp',
    'ports': ['22623']
  }],
  'sourceTags': [
    context.properties['infra_id'] + '-master',
    context.properties['infra_id'] + '-worker'
  ],
  'targetTags': [context.properties['infra_id'] + '-master']
}
}, {
'name': context.properties['infra_id'] + '-internal-network',
'type': 'compute.v1.firewall',
'properties': {
  'network': context.properties['cluster_network'],
  'allowed': [{
    'IPProtocol': 'icmp'
  },{
    'IPProtocol': 'tcp',
    'ports': ['22']
  }],
  'sourceRanges': [context.properties['network_cidr']],
  'targetTags': [
    context.properties['infra_id'] + '-master',
    context.properties['infra_id'] + '-worker'
  ]
}
}, {
'name': context.properties['infra_id'] + '-internal-cluster',
'type': 'compute.v1.firewall',
'properties': {
  'network': context.properties['cluster_network'],
  'allowed': [{
    'IPProtocol': 'udp',
    'ports': ['4789', '6081']
  },{
    'IPProtocol': 'tcp',
    'ports': ['9000-9999']
  },{
    'IPProtocol': 'udp',
    'ports': ['9000-9999']
  },{
    'IPProtocol': 'tcp',
    'ports': ['10250']
  },{
    'IPProtocol': 'tcp',

```

```

      'ports': ['30000-32767']
    },{
      'IPProtocol': 'udp',
      'ports': ['30000-32767']
    }],
    'sourceTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ],
    'targetTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ]
  }
}
}
}

return {'resources': resources}

```

4.9.11. Creating IAM roles in GCP

You must create IAM roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.
2. Create a **03_iam.yaml** resource definition file:

```

$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py

```

```
properties:
  infra_id: '${INFRA_ID}' 1
EOF
```

- 1** `infra_id` is the `INFRA_ID` infrastructure name from the extraction step.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=$(gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '[0].email')
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=$(gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email')
```

6. Export the variable for the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=$(gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink')
```

7. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. Create a service account key and store it locally for later use:

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

4.9.11.1. Deployment Manager template for IAM roles

You can use the following Deployment Manager template to deploy the IAM roles that you need for your OpenShift Container Platform cluster:

Example 4.6. 03_iam.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }
    ]

    return {'resources': resources}
```

4.9.12. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS image from the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The file name contains the OpenShift Container Platform version number in the format **rhcos-
<version>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

```
$ gsutil mb gs://<bucket_name>
```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz`
```

5. Create the cluster image:

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
--source-uri="${IMAGE_SOURCE}"
```

4.9.13. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Ensure pyOpenSSL is installed.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.
2. Export the location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. Create a bucket and upload the **bootstrap.ign** file:

■

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. Create a **04_bootstrap.yaml** resource definition file:

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2 **region** is the region to deploy the cluster into, for example **us-central1**.
- 3 **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-b**.
- 4 **cluster_network** is the **selfLink** URL to the cluster network.
- 5 **control_subnet** is the **selfLink** URL to the control subnet.
- 6 **image** is the **selfLink** URL to the RHCOS image.
- 7 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 8 **root_volume_size** is the boot disk size for the bootstrap machine.
- 9 **bootstrap_ign** is the URL output when creating a signed URL.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the bootstrap machine manually.

- a. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances \
  ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
bootstrap
```

- b. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend \
  ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
```

4.9.13.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 4.7. 04_bootstrap.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"},"verification":{}}},"timeouts":{},"version":"2.1.0"},"network":{},"passwd":{},"storage":
{},"systemd":{}}',
                }
            ]
        }
    ]
```

```

    },
    'networkInterfaces': [{
      'subnetwork': context.properties['control_subnet'],
      'accessConfigs': [{
        'natIP': '${ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address}'
      }]
    }],
    'tags': {
      'items': [
        context.properties['infra_id'] + '-master',
        context.properties['infra_id'] + '-bootstrap'
      ]
    },
    'zone': context.properties['zone']
  }
}, {
  'name': context.properties['infra_id'] + '-bootstrap-instance-group',
  'type': 'compute.v1.instanceGroup',
  'properties': {
    'namedPorts': [
      {
        'name': 'ignition',
        'port': 22623
      }, {
        'name': 'https',
        'port': 6443
      }
    ],
    'network': context.properties['cluster_network'],
    'zone': context.properties['zone']
  }
}]

return {'resources': resources}

```

4.9.14. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use. One way to create these machines is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.
2. Export the following variable required by the resource definition:

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. Create a **05_control_plane.yaml** resource definition file:

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2 **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.
- 3 **control_subnet** is the **selfLink** URL to the control subnet.
- 4 **image** is the **selfLink** URL to the RHCOS image.
- 5 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 6 **service_account_email** is the email address for the master service account that you created.

7 **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-m-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-m-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-m-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

4.9.14.1. Deployment Manager template for control plane machines

You can use the following Deployment Manager template to deploy the control plane machines that you need for your OpenShift Container Platform cluster:

Example 4.8. 05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-m-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }
        ],
        'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
```

```

context.properties['machine_type'],
  'metadata': {
    'items': [{
      'key': 'user-data',
      'value': context.properties['ignition']
    }]
  },
  'networkInterfaces': [{
    'subnetwork': context.properties['control_subnet']
  }],
  'serviceAccounts': [{
    'email': context.properties['service_account_email'],
    'scopes': ['https://www.googleapis.com/auth/cloud-platform']
  }],
  'tags': {
    'items': [
      context.properties['infra_id'] + '-master',
    ]
  },
  'zone': context.properties['zones'][0]
}
}, {
  'name': context.properties['infra_id'] + '-m-1',
  'type': 'compute.v1.instance',
  'properties': {
    'disks': [{
      'autoDelete': True,
      'boot': True,
      'initializeParams': {
        'diskSizeGb': context.properties['root_volume_size'],
        'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
        'sourceImage': context.properties['image']
      }
    }],
    'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
    'metadata': {
      'items': [{
        'key': 'user-data',
        'value': context.properties['ignition']
      }]
    },
    'networkInterfaces': [{
      'subnetwork': context.properties['control_subnet']
    }],
    'serviceAccounts': [{
      'email': context.properties['service_account_email'],
      'scopes': ['https://www.googleapis.com/auth/cloud-platform']
    }],
    'tags': {
      'items': [
        context.properties['infra_id'] + '-master',
      ]
    },
    'zone': context.properties['zones'][1]
  }
}

```

```

    }, {
      'name': context.properties['infra_id'] + '-m-2',
      'type': 'compute.v1.instance',
      'properties': {
        'disks': [{
          'autoDelete': True,
          'boot': True,
          'initializeParams': {
            'diskSizeGb': context.properties['root_volume_size'],
            'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
            'sourceImage': context.properties['image']
          }
        }
      ],
      'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
      'metadata': {
        'items': [{
          'key': 'user-data',
          'value': context.properties['ignition']
        }
      ]
    },
    'networkInterfaces': [{
      'subnetwork': context.properties['control_subnet']
    }],
    'serviceAccounts': [{
      'email': context.properties['service_account_email'],
      'scopes': ['https://www.googleapis.com/auth/cloud-platform']
    }],
    'tags': {
      'items': [
        context.properties['infra_id'] + '-master',
      ]
    },
    'zone': context.properties['zones'][2]
  }
]

return {'resources': resources}

```

4.9.15. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

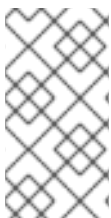
2. Delete the bootstrap resources:

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

4.9.16. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.



NOTE

If you do not use the provided Deployment Manager template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.
2. Export the variables that the resource definition uses.

- a. Export the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

- b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email`)
```

- c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' 1
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 2
    zone: '${ZONE_0}' 3
    compute_subnet: '${COMPUTE_SUBNET}' 4
    image: '${CLUSTER_IMAGE}' 5
    machine_type: 'n1-standard-4' 6
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
```

```

compute_subnet: '${COMPUTE_SUBNET}' 11
image: '${CLUSTER_IMAGE}' 12
machine_type: 'n1-standard-4' 13
root_volume_size: '128'
service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
ignition: '${WORKER_IGNITION}' 15
EOF

```

- 1 **name** is the name of the worker machine, for example **worker-0**.
- 2 9 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 3 10 **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.
- 4 11 **compute_subnet** is the **selfLink** URL to the compute subnet.
- 5 12 **image** is the **selfLink** URL to the RHCOS image.
- 6 13 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 7 14 **service_account_email** is the email address for the worker service account that you created.
- 8 15 **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.
5. Create the deployment by using the **gcloud** CLI:

```

$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml

```

4.9.16.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 4.9. 06_worker.py Deployment Manager template

```

def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }
        ]
    }
    ],

```

```

        'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['compute_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-worker',
            ]
        },
        'zone': context.properties['zone']
    }
}
]]

return {'resources': resources}

```

4.9.17. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.9.17.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```


5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.9.17.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.9.17.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.9.18. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.9.19. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```

NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 63m  v1.18.3
master-1  Ready   master 63m  v1.18.3
master-2  Ready   master 64m  v1.18.3
worker-0  NotReady worker 76s  v1.18.3
worker-1  NotReady worker 70s  v1.18.3

```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE  REQUESTOR                                     CONDITION
csr-8b2br 15m  system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m  system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

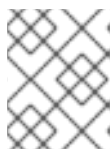
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready    master   73m   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

4.9.20. Optional: Adding the ingress DNS records

If you removed the DNS zone configuration when creating Kubernetes manifests and generating Ignition configs, you must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}**. or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.
- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

```
$ oc -n openshift-ingress get service router-default
```

Example output

```
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
router-default LoadBalancer  172.30.18.154 35.233.157.184 80:32288/TCP,443:31215/TCP 98
```

2. Add the A record to your zones:

- To use A records:
 - i. Export the variable for the router IP address:

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
```

- ii. Add the A record to the private zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

- iii. For an external cluster, also add the A record to the public zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- To add explicit domains instead of using a wildcard, create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

4.9.21. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned GCP infrastructure.
- Install the **oc** CLI and log in.

Procedure

1. Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Observe the running state of your cluster.

a. Run the following command to view the current cluster version and status:

```
$ oc get clusterversion
```

Example output

```
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
version   False    True       24m         Working towards 4.5.4: 99% complete
```

b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

```
$ oc get clusteroperators
```

Example output

```
NAME                               VERSION  AVAILABLE  PROGRESSING  DEGRADED  SINCE
authentication                      4.5.4   True       False        False     7m56s
cloud-credential                    4.5.4   True       False        False     31m
cluster-autoscaler                   4.5.4   True       False        False     16m
console                             4.5.4   True       False        False     10m
csi-snapshot-controller              4.5.4   True       False        False     16m
dns                                  4.5.4   True       False        False     22m
etcd                                 4.5.4   False      False        False     25s
image-registry                       4.5.4   True       False        False     16m
ingress                              4.5.4   True       False        False     16m
insights                             4.5.4   True       False        False     17m
kube-apiserver                       4.5.4   True       False        False     19m
kube-controller-manager              4.5.4   True       False        False     20m
kube-scheduler                       4.5.4   True       False        False     20m
kube-storage-version-migrator        4.5.4   True       False        False     16m
machine-api                         4.5.4   True       False        False     22m
machine-config                       4.5.4   True       False        False     22m
marketplace                          4.5.4   True       False        False     16m
monitoring                           4.5.4   True       False        False     10m
network                              4.5.4   True       False        False     23m
node-tuning                          4.5.4   True       False        False     23m
openshift-apiserver                  4.5.4   True       False        False     17m
openshift-controller-manager         4.5.4   True       False        False     15m
openshift-samples                    4.5.4   True       False        False     16m
```

operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

- c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE                               NAME
READY  STATUS  RESTARTS  AGE
kube-system                               etcd-member-ip-10-0-3-111.us-east-
2.compute.internal                       1/1    Running  0    35m
kube-system                               etcd-member-ip-10-0-3-239.us-east-
2.compute.internal                       1/1    Running  0    37m
kube-system                               etcd-member-ip-10-0-3-24.us-east-
2.compute.internal                       1/1    Running  0    35m
openshift-apiserver-operator             openshift-apiserver-operator-6d6674f4f4-
h7t2t                                   1/1    Running  1    37m
openshift-apiserver                      apiserver-fm48r
1/1    Running  0    30m
openshift-apiserver                      apiserver-fxkvv
1/1    Running  0    29m
openshift-apiserver                      apiserver-q85nm
1/1    Running  0    29m
...
openshift-service-ca-operator            openshift-service-ca-operator-66ff6dc6cd-
9r257                                   1/1    Running  0    37m
openshift-service-ca                    apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1    Running  0    35m
openshift-service-ca                    configmap-cabundle-injector-8498544d7-
25qn6                                   1/1    Running  0    35m
openshift-service-ca                    service-serving-cert-signer-6445fc9c6-wqdqn
1/1    Running  0    35m
openshift-service-catalog-apiserver-operator  openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w             1/1    Running  0    32m
openshift-service-catalog-controller-manager-operator  openshift-service-catalog-
controller-manager-operator-b78cr2lnm  1/1    Running  0    31m

```

When the current cluster version is **AVAILABLE**, the installation is complete.

4.9.22. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

4.10. INSTALLING A CLUSTER WITH SHARED VPC ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES

In OpenShift Container Platform version 4.5, you can install a cluster into a shared Virtual Private Cloud (VPC) on Google Cloud Platform (GCP) that uses infrastructure that you provide. In this context, a cluster installed into a shared VPC is a cluster that is configured to use a VPC from a project different from where the cluster is being deployed.

A shared VPC enables an organization to connect resources from multiple projects to a common VPC network. You can communicate within the organization securely and efficiently by using internal IPs from that network. For more information about shared VPC, see [Shared VPC overview](#) in the GCP documentation.

The steps for performing a user-provided infrastructure installation into a shared VPC are outlined here. Several [Deployment Manager](#) templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods.

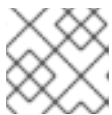


IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

4.10.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.



NOTE

Be sure to also review this site list if you are configuring a proxy.

4.10.2. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

4.10.3. Configuring the GCP project that hosts your cluster

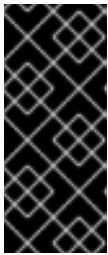
Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

4.10.3.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>.<base_domain>** URL; the Premium Tier is required for internal load balancing.

4.10.3.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See [Enabling services](#) in the GCP documentation.

Table 4.33. Required API services

API service	Console service name
Cloud Deployment Manager V2 API	deploymentmanager.googleapis.com
Compute Engine API	compute.googleapis.com
Google Cloud APIs	cloudapis.googleapis.com
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com
IAM Service Account Credentials API	iamcredentials.googleapis.com

API service	Console service name
Identity and Access Management (IAM) API	iam.googleapis.com
Service Management API	servicemanagement.googleapis.com
Service Usage API	serviceusage.googleapis.com
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

4.10.3.3. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default [Quotas](#) do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 4.34. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Networking	Global	11	1
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0

**NOTE**

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europa-north1**
- **europa-west2**
- **europa-west3**
- **europa-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

4.10.3.4. Creating a service account in GCP

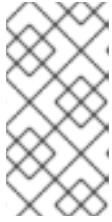
OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See [Creating a service account](#) in the GCP documentation.
2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).

**NOTE**

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See [Creating service account keys](#) in the GCP documentation.
The service account key is required to create a cluster.

4.10.3.4.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Security Admin
- Service Account Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor
- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 4.35. GCP service account permissions

Account	Roles
Control Plane	roles/compute.instanceAdmin

Account	Roles
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin

4.10.3.5. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europa-north1** (Hamina, Finland)
- **europa-west1** (St. Ghislain, Belgium)
- **europa-west2** (London, England, UK)
- **europa-west3** (Frankfurt, Germany)
- **europa-west4** (Eemshaven, Netherlands)
- **europa-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)

- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.10.3.6. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.
- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **\$PATH**:

- **gcloud**
- **gsutil**

See [Install the latest Cloud SDK version](#) in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.
See [Authorizing with a service account](#) in the GCP documentation.

4.10.4. Configuring the GCP project that hosts your shared VPC network

If you use a shared Virtual Private Cloud (VPC) to host your OpenShift Container Platform cluster in Google Cloud Platform (GCP), you must configure the project that hosts it.



NOTE

If you already have a project that hosts the shared VPC network, review this section to ensure that the project meets all of the requirements to install an OpenShift Container Platform cluster.

Procedure

1. Create a project to host the shared VPC for your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.
2. Create a service account in the project that hosts your shared VPC. See [Creating a service account](#) in the GCP documentation.
3. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).

**NOTE**

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

The service account for the project that hosts the shared VPC network requires the following roles:

- Compute Network User
- Compute Security Admin
- Deployment Manager Editor
- DNS Administrator
- Security Admin
- Network Management Admin

4.10.4.1. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the project that hosts the shared VPC that you install the cluster into. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.

**NOTE**

If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see [Google Domains](#).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](#) in the GCP documentation.
Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
3. Extract the new authoritative name servers from the hosted zone records. See [Look up your Cloud DNS name servers](#) in the GCP documentation.
You typically have four name servers.
4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: [How to switch to custom name servers](#).
5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.

6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

4.10.4.2. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC** section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.
2. Export the following variables required by the resource definition:

- a. Export the control plane CIDR:

```
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
```

- b. Export the compute CIDR:

```
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'
```

- c. Export the region to deploy the VPC network and cluster to:

```
$ export REGION='<region>'
```

3. Export the variable for the ID of the project that hosts the shared VPC:

```
$ export HOST_PROJECT=<host_project>
```

4. Export the variable for the email of the service account that belongs to host project:

```
$ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
```

5. Create a **01_vpc.yaml** resource definition file:

```
$ cat <<EOF >01_vpc.yaml
imports:
```

```

- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '<prefix>' ❶
    region: '${REGION}' ❷
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' ❸
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' ❹
EOF

```

- ❶ **infra_id** is the prefix of the network name.
- ❷ **region** is the region to deploy the cluster into, for example **us-central1**.
- ❸ **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.
- ❹ **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config
01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} ❶
```

- ❶ For **<vpc_deployment_name>**, specify the name of the VPC to deploy.

7. Export the VPC variable that other components require:

- a. Export the name of the host project network:

```
$ export HOST_PROJECT_NETWORK=<vpc_network>
```

- b. Export the name of the host project control plane subnet:

```
$ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
```

- c. Export the name of the host project compute subnet:

```
$ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
```

8. Set up the shared VPC. See [Setting up Shared VPC](#) in the GCP documentation.

4.10.4.2.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

Example 4.10. 01_vpc.py Deployment Manager template

```
def GenerateConfig(context):
```

```

resources = [{
  'name': context.properties['infra_id'] + '-network',
  'type': 'compute.v1.network',
  'properties': {
    'region': context.properties['region'],
    'autoCreateSubnetworks': False
  }
}, {
  'name': context.properties['infra_id'] + '-master-subnet',
  'type': 'compute.v1.subnetwork',
  'properties': {
    'region': context.properties['region'],
    'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
    'ipCidrRange': context.properties['master_subnet_cidr']
  }
}, {
  'name': context.properties['infra_id'] + '-worker-subnet',
  'type': 'compute.v1.subnetwork',
  'properties': {
    'region': context.properties['region'],
    'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
    'ipCidrRange': context.properties['worker_subnet_cidr']
  }
}, {
  'name': context.properties['infra_id'] + '-router',
  'type': 'compute.v1.router',
  'properties': {
    'region': context.properties['region'],
    'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
    'nats': [{
      'name': context.properties['infra_id'] + '-nat-master',
      'natIpAllocateOption': 'AUTO_ONLY',
      'minPortsPerVm': 7168,
      'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
      'subnetworks': [{
        'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
        'sourceIpRangesToNat': ['ALL_IP_RANGES']
      }]
    }]
  }, {
    'name': context.properties['infra_id'] + '-nat-worker',
    'natIpAllocateOption': 'AUTO_ONLY',
    'minPortsPerVm': 512,
    'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
    'subnetworks': [{
      'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
      'sourceIpRangesToNat': ['ALL_IP_RANGES']
    }]
  }
}]
}
return {'resources': resources}

```

4.10.5. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files.

4.10.5.1. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

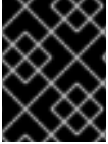


IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

4.10.5.2. Sample customized install-config.yaml file for GCP

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.



IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it.

```

apiVersion: v1
baseDomain: example.com 1
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
compute: 5
- hyperthreading: Enabled 6
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 0
metadata:
  name: test-cluster
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 7
    region: us-central1 8
pullSecret: '{"auths": ...}'
fips: false 9
sshKey: ssh-ed25519 AAAA... 10
publish: Internal 11

```

- 1 Specify the public DNS on the host project.
- 2 5 If you do not provide these parameters and values, the installation program provides the default value.
- 3 6 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen - and the first line of the **controlPlane** section must not. Although both

must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

- 4 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Use larger machine types, such as **n1-standard-8**, for your machines if you disable simultaneous multithreading.

- 7 Specify the main project where the VM instances reside.
- 8 Specify the region that your VPC network is in.
- 9 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 10 You can optionally provide the **sshKey** value that you use to access the machines in your cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 11 How to publish the user-facing endpoints of your cluster. Set **publish** to **Internal** to deploy a private cluster, which cannot be accessed from the Internet. The default value is **External**. To use a shared VPC in a cluster that uses infrastructure that you provision, you must set **publish** to **Internal**. The installation program will no longer be able to access the public DNS zone for the base domain in the host project.

4.10.5.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

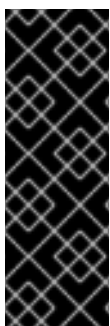
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

4.10.5.4. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:

- Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
- Locate the `mastersSchedulable` parameter and set its value to **False**.
- Save and exit the file.

- Remove the `privateZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
  id: mycluster-100419-private-zone
status: {}
```

- ❶ Remove this section completely.

- Configure the cloud provider for your VPC.

- Open the `<installation_directory>/manifests/cloud-provider-config.yaml` file.
- Add the `network-project-id` parameter and set its value to the ID of project that hosts the shared VPC network.
- Add the `network-name` parameter and set its value to the name of the shared VPC network that hosts the OpenShift Container Platform cluster.
- Replace the value of the `subnetwork-name` parameter with the value of the shared VPC subnet that hosts your compute machines.

The contents of the `<installation_directory>/manifests/cloud-provider-config.yaml` resemble the following example:

```

config: |+
  [global]
  project-id      = example-project
  regional       = true
  multizone      = true
  node-tags      = opensh-ptzzx-master
  node-tags      = opensh-ptzzx-worker
  node-instance-prefix = opensh-ptzzx
  external-instance-groups-prefix = opensh-ptzzx
  network-project-id = example-shared-vpc
  network-name    = example-network
  subnetwork-name = example-worker-subnet

```

7. If you deploy a cluster that is not on a private network, open the `<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml` file and replace the value of the `scope` parameter with `External`. The contents of the file resemble the following example:

```

apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
      type: LoadBalancerService
status:
  availableReplicas: 0
  domain: ""
  selector: ""

```

8. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> ❶
```

- ❶ For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:

```

.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign

```

4.10.6. Exporting common variables

4.10.6.1. Extracting the infrastructure name

Additional resources

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

4.10.6.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).



NOTE

Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

1. Export the following common variables to be used by the provided Deployment Manager templates:

```
$ export BASE_DOMAIN='<base_domain>' 1
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>' 2
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 3
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
```

- 1 2 Supply the values for the host project.
- 3 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

4.10.7. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 4.36. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 4.37. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 4.38. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer:** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 4.39. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
------	----------------------------------	----------	----------	-------------

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

- Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 4.40. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

4.10.8. Creating load balancers in GCP

You must configure load balancers in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.

**NOTE**

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.
2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.
3. Export the variables that the deployment template uses:
 - a. Export the cluster network location:

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe
${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`)
```

- b. Export the control plane subnet location:

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

- c. Export the three zones that the cluster uses:

```
$ export ZONE_0=`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`
```

```
$ export ZONE_1=`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`
```

```
$ export ZONE_2=`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`
```

4. Create a **02_infra.yaml** resource definition file:

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py ❶
resources:
- name: cluster-lb-ext ❷
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' ❸
    region: '${REGION}' ❹
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' ❺
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: ❻
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

❶ ❷ Required only when deploying an external cluster.

❸ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

❹ **region** is the region to deploy the cluster into, for example **us-central1**.

❺ **control_subnet** is the URI to the control subnet.

❻ **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:


```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --
region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-
public-ip --region=${REGION} --format json | jq -r .address`)
```

4.10.8.1. Deployment Manager template for the external load balancer

You can use the following Deployment Manager template to deploy the external load balancer that you need for your OpenShift Container Platform cluster:

Example 4.11. 02_lb_ext.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }
    ]

    return {'resources': resources}
```

4.10.8.2. Deployment Manager template for the internal load balancer

You can use the following Deployment Manager template to deploy the internal load balancer that you need for your OpenShift Container Platform cluster:

Example 4.12. `02_lb_int.py` Deployment Manager template

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
            '.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
            check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
            service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
```

```

        'ports': ['6443', '22623'],
        'region': context.properties['region'],
        'subnetwork': context.properties['control_subnet']
    }
}
}}

for zone in context.properties['zones']:
    resources.append({
        'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': zone
        }
    })

return {'resources': resources}

```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

4.10.9. Creating a private DNS zone in GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.
2. Create a **02_dns.yaml** resource definition file:

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.
- 3** **cluster_network** is the **selfLink** URL to the cluster network.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:
 - a. Add the internal DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

- b. For an external cluster, also add the external DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

4.10.9.1. Deployment Manager template for the private DNS

You can use the following Deployment Manager template to deploy the private DNS that you need for your OpenShift Container Platform cluster:

Example 4.13. `02_dns.py` Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

4.10.10. Creating firewall rules in GCP

You must create firewall rules in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.
2. Create a **03_firewall.yaml** resource definition file:

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' 1
    infra_id: '${INFRA_ID}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
    network_cidr: '${NETWORK_CIDR}' 4
EOF
```

- 1 **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **\${NETWORK_CIDR}**.
- 2 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 3 **cluster_network** is the **selfLink** URL to the cluster network.
- 4 **network_cidr** is the CIDR of the VPC network, for example **10.0.0.0/16**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4.10.10.1. Deployment Manager template for firewall rules

You can use the following Deployment Manager template to deploy the firewall rules that you need for your OpenShift Container Platform cluster:

Example 4.14. 03_firewall.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
        }
    ]
```

```

    'sourceRanges': [context.properties['allowed_external_cidr']],
    'targetTags': [context.properties['infra_id'] + '-bootstrap']
  }
}, {
  'name': context.properties['infra_id'] + '-api',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [{
      'IPProtocol': 'tcp',
      'ports': ['6443']
    }],
    'sourceRanges': [context.properties['allowed_external_cidr']],
    'targetTags': [context.properties['infra_id'] + '-master']
  }
}, {
  'name': context.properties['infra_id'] + '-health-checks',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [{
      'IPProtocol': 'tcp',
      'ports': ['6080', '6443', '22624']
    }],
    'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
    'targetTags': [context.properties['infra_id'] + '-master']
  }
}, {
  'name': context.properties['infra_id'] + '-etcd',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [{
      'IPProtocol': 'tcp',
      'ports': ['2379-2380']
    }],
    'sourceTags': [context.properties['infra_id'] + '-master'],
    'targetTags': [context.properties['infra_id'] + '-master']
  }
}, {
  'name': context.properties['infra_id'] + '-control-plane',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [{
      'IPProtocol': 'tcp',
      'ports': ['10257']
    }],
    {
      'IPProtocol': 'tcp',
      'ports': ['10259']
    },
    {
      'IPProtocol': 'tcp',
      'ports': ['22623']
    }
  ],
  'sourceTags': [
    context.properties['infra_id'] + '-master',

```

```

        context.properties['infra_id'] + '-worker'
    ],
    'targetTags': [context.properties['infra_id'] + '-master']
  }
}, {
  'name': context.properties['infra_id'] + '-internal-network',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [{
      'IPProtocol': 'icmp'
    }, {
      'IPProtocol': 'tcp',
      'ports': ['22']
    }],
    'sourceRanges': [context.properties['network_cidr']],
    'targetTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ]
  }
}, {
  'name': context.properties['infra_id'] + '-internal-cluster',
  'type': 'compute.v1.firewall',
  'properties': {
    'network': context.properties['cluster_network'],
    'allowed': [{
      'IPProtocol': 'udp',
      'ports': ['4789', '6081']
    }, {
      'IPProtocol': 'tcp',
      'ports': ['9000-9999']
    }, {
      'IPProtocol': 'udp',
      'ports': ['9000-9999']
    }, {
      'IPProtocol': 'tcp',
      'ports': ['10250']
    }, {
      'IPProtocol': 'tcp',
      'ports': ['30000-32767']
    }, {
      'IPProtocol': 'udp',
      'ports': ['30000-32767']
    }],
    'sourceTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ],
    'targetTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ]
  }
}

```



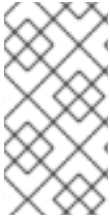
```

}}
return {'resources': resources}

```

4.10.11. Creating IAM roles in GCP

You must create IAM roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.
2. Create a **03_iam.yaml** resource definition file:

```

$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' 1
EOF

```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3. Create the deployment by using the **gcloud** CLI:

```

$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml

```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=$(gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '[0].email')
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=$(gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email')
```

6. Assign the permissions that the installation program requires to the service accounts for the subnets that host the control plane and compute subnets:

- a. Grant the **networkViewer** role of the project that hosts your shared VPC to the master service account:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
projects add-iam-policy-binding ${HOST_PROJECT} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
"roles/compute.networkViewer"
```

- b. Grant the **networkUser** role to the master service account for the control plane subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

- c. Grant the **networkUser** role to the worker service account for the control plane subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

- d. Grant the **networkUser** role to the master service account for the compute subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

- e. Grant the **networkUser** role to the worker service account for the compute subnet:

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

7. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

```

8. Create a service account key and store it locally for later use:

```

$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}

```

4.10.11.1. Deployment Manager template for IAM roles

You can use the following Deployment Manager template to deploy the IAM roles that you need for your OpenShift Container Platform cluster:

Example 4.15. `03_iam.py` Deployment Manager template

```

def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }
    ]

    return {'resources': resources}

```

4.10.12. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS image from the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The file name contains the OpenShift Container Platform version number in the format **rhcos-
<version>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

```
$ gsutil mb gs://<bucket_name>
```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz  
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

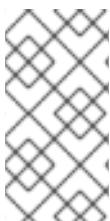
```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-  
gcp.x86_64.tar.gz`
```

5. Create the cluster image:

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \  
--source-uri="${IMAGE_SOURCE}"
```

4.10.13. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.

- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Ensure pyOpenSSL is installed.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.
2. Export the location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=$(gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. Create a bucket and upload the **bootstrap.ign** file:

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-
bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}`
```

5. Create a **04_bootstrap.yaml** resource definition file:

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2 **region** is the region to deploy the cluster into, for example **us-central1**.
- 3 **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-b**.
- 4 **cluster_network** is the **selfLink** URL to the cluster network.
- 5 **control_subnet** is the **selfLink** URL to the control subnet.
- 6 **image** is the **selfLink** URL to the RHCOS image.
- 7 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 8 **root_volume_size** is the boot disk size for the bootstrap machine.
- 9 **bootstrap_ign** is the URL output when creating a signed URL.

6. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-
instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal-backend-service
--region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --instance-
group-zone=${ZONE_0}
```

4.10.13.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 4.16. 04_bootstrap.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
```

```

        'autoDelete': True,
        'boot': True,
        'initializeParams': {
            'diskSizeGb': context.properties['root_volume_size'],
            'sourceImage': context.properties['image']
        }
    }],
    'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
    'metadata': {
        'items': [{
            'key': 'user-data',
            'value': '{"ignition":{"config":{"replace":{"source":"" + context.properties['bootstrap_ign']
+ ""},"verification":{}}},"timeouts":{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":
{},"systemd":{}}',
        }]
    },
    'networkInterfaces': [{
        'subnetwork': context.properties['control_subnet'],
        'accessConfigs': [{
            'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
        }]
    }],
    'tags': {
        'items': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-bootstrap'
        ]
    },
    'zone': context.properties['zone']
}
}, {
    'name': context.properties['infra_id'] + '-bootstrap-instance-group',
    'type': 'compute.v1.instanceGroup',
    'properties': {
        'namedPorts': [
            {
                'name': 'ignition',
                'port': 22623
            }, {
                'name': 'https',
                'port': 6443
            }
        ],
        'network': context.properties['cluster_network'],
        'zone': context.properties['zone']
    }
}
]]

return {'resources': resources}

```

4.10.14. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use. One way to create these machines is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.
2. Export the following variable required by the resource definition:

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. Create a **05_control_plane.yaml** resource definition file:

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6
```



```
ignition: '${MASTER_IGNITION}' 7
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2 **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.
- 3 **control_subnet** is the **selfLink** URL to the control subnet.
- 4 **image** is the **selfLink** URL to the RHCOS image.
- 5 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 6 **service_account_email** is the email address for the master service account that you created.
- 7 **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.
 - Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-m-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-m-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-m-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

4.10.14.1. Deployment Manager template for control plane machines

You can use the following Deployment Manager template to deploy the control plane machines that you need for your OpenShift Container Platform cluster:

Example 4.17. 05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):
```

```

resources = [{
  'name': context.properties['infra_id'] + '-m-0',
  'type': 'compute.v1.instance',
  'properties': {
    'disks': [{
      'autoDelete': True,
      'boot': True,
      'initializeParams': {
        'diskSizeGb': context.properties['root_volume_size'],
        'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
        'sourceImage': context.properties['image']
      }
    }
  ],
  'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
  'metadata': {
    'items': [{
      'key': 'user-data',
      'value': context.properties['ignition']
    }
  ]
},
  'networkInterfaces': [{
    'subnetwork': context.properties['control_subnet']
  }],
  'serviceAccounts': [{
    'email': context.properties['service_account_email'],
    'scopes': ['https://www.googleapis.com/auth/cloud-platform']
  }],
  'tags': {
    'items': [
      context.properties['infra_id'] + '-master',
    ]
  },
  'zone': context.properties['zones'][0]
}
], {
  'name': context.properties['infra_id'] + '-m-1',
  'type': 'compute.v1.instance',
  'properties': {
    'disks': [{
      'autoDelete': True,
      'boot': True,
      'initializeParams': {
        'diskSizeGb': context.properties['root_volume_size'],
        'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
        'sourceImage': context.properties['image']
      }
    }
  ],
  'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
  'metadata': {
    'items': [{
      'key': 'user-data',
      'value': context.properties['ignition']
    }
  ]
}
]
}

```

```

    }
  },
  'networkInterfaces': [{
    'subnetwork': context.properties['control_subnet']
  }],
  'serviceAccounts': [{
    'email': context.properties['service_account_email'],
    'scopes': ['https://www.googleapis.com/auth/cloud-platform']
  }],
  'tags': {
    'items': [
      context.properties['infra_id'] + '-master',
    ]
  },
  'zone': context.properties['zones'][1]
}
}, {
  'name': context.properties['infra_id'] + '-m-2',
  'type': 'compute.v1.instance',
  'properties': {
    'disks': [{
      'autoDelete': True,
      'boot': True,
      'initializeParams': {
        'diskSizeGb': context.properties['root_volume_size'],
        'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
        'sourceImage': context.properties['image']
      }
    }
  ]],
  'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
  'metadata': {
    'items': [{
      'key': 'user-data',
      'value': context.properties['ignition']
    }
  ]
},
  'networkInterfaces': [{
    'subnetwork': context.properties['control_subnet']
  }],
  'serviceAccounts': [{
    'email': context.properties['service_account_email'],
    'scopes': ['https://www.googleapis.com/auth/cloud-platform']
  }],
  'tags': {
    'items': [
      context.properties['infra_id'] + '-master',
    ]
  },
  'zone': context.properties['zones'][2]
}
]]
return {'resources': resources}

```

4.10.15. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level info 2
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

2. Delete the bootstrap resources:

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

4.10.16. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.



NOTE

If you do not use the provided Deployment Manager template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.
2. Export the variables that the resource definition uses.
 - a. Export the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

- b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email`)
```

- c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' 1
```

```

type: 06_worker.py
properties:
  infra_id: '${INFRA_ID}' 2
  zone: '${ZONE_0}' 3
  compute_subnet: '${COMPUTE_SUBNET}' 4
  image: '${CLUSTER_IMAGE}' 5
  machine_type: 'n1-standard-4' 6
  root_volume_size: '128'
  service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
  ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF

```

- 1 **name** is the name of the worker machine, for example **worker-0**.
- 2 9 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 3 10 **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.
- 4 11 **compute_subnet** is the **selfLink** URL to the compute subnet.
- 5 12 **image** is the **selfLink** URL to the RHCOS image.
- 6 13 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 7 14 **service_account_email** is the email address for the worker service account that you created.
- 8 15 **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.
5. Create the deployment by using the **gcloud** CLI:

```

$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml

```

4.10.16.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 4.18. `06_worker.py` Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    ]

    return {'resources': resources}
```

4.10.17. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

4.10.17.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.10.17.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

4.10.17.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

4.10.18. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.10.19. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready     master   63m   v1.18.3
master-1  Ready     master   63m   v1.18.3
master-2  Ready     master   64m   v1.18.3
worker-0  NotReady  worker   76s   v1.18.3
worker-1  NotReady  worker   70s   v1.18.3
```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper  Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

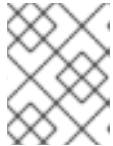
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready    master   73m   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

4.10.20. Adding the ingress DNS records

DNS zone configuration is removed when creating Kubernetes manifests and generating Ignition configs. You must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}**. or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.
- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

```
$ oc -n openshift-ingress get service router-default
```

Example output

```
-
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
router-default	LoadBalancer	172.30.18.154	35.233.157.184	80:32288/TCP,443:31215/TCP	98

2. Add the A record to your zones:

- To use A records:

- Export the variable for the router IP address:

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
```

- Add the A record to the private zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

- For an external cluster, also add the A record to the public zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME} -
-project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
```

- To add explicit domains instead of using a wildcard, create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

4.10.21. Adding ingress firewall rules

The cluster requires several firewall rules. If you do not use a shared VPC, these rules are created by the ingress controller via the GCP cloud provider. When you use a shared VPC, you can either create cluster-wide firewall rules for all services now or create each rule based on events, when the cluster requests access. By creating each rule when the cluster requests access, you know exactly which firewall rules are required. By creating cluster-wide firewall rules, you can apply the same rule set across multiple clusters.

If you choose to create each rule based on events, you must create firewall rules after you provision the cluster and during the life of the cluster when the console notifies you that rules are missing. Events that are similar to the following event are displayed, and you must add the firewall rules that are required:

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

Example output

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-
a26e631036a3f46cba28f8df67266d55 --network example-network --description "
{"kubernetes.io/service-name":"openshift-ingress/router-default", "kubernetes.io/service-
ip":"35.237.236.234"}" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exampl-fqzq7-
master,exampl-fqzq7-worker --project example-project`
```

If you encounter issues when creating these rule-based events, you can configure the cluster-wide firewall rules while your cluster is running.

4.10.21.1. Creating cluster-wide firewall rules for a shared VPC in GCP

You can create cluster-wide firewall rules to allow the access that the OpenShift Container Platform cluster requires.



WARNING

If you do not choose to create firewall rules based on cluster events, you must create cluster-wide firewall rules.

Prerequisites

- You exported the variables that the Deployment Manager templates require to deploy your cluster.
- You created the networking and load balancing components in GCP that your cluster requires.

Procedure

1. Add a single firewall rule to allow the Google Cloud Engine health checks to access all of the services. This rule enables the ingress load balancers to determine the health status of their instances.

```
$ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --
network="{CLUSTER_NETWORK}" --source-
ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-
tags="{INFRA_ID}-master,{INFRA_ID}-worker" {INFRA_ID}-ingress-hc --
account={HOST_PROJECT_ACCOUNT} --project={HOST_PROJECT}
```

2. Add a single firewall rule to allow access to all cluster services:

- For an external cluster:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="{CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-
tags="{INFRA_ID}-master,{INFRA_ID}-worker" {INFRA_ID}-ingress --
account={HOST_PROJECT_ACCOUNT} --project={HOST_PROJECT}
```

- For a private cluster:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="{CLUSTER_NETWORK}" --source-ranges={NETWORK_CIDR} --target-
tags="{INFRA_ID}-master,{INFRA_ID}-worker" {INFRA_ID}-ingress --
account={HOST_PROJECT_ACCOUNT} --project={HOST_PROJECT}
```

Because this rule only allows traffic on TCP ports **80** and **443**, ensure that you add all the ports that your services use.

4.10.22. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned GCP infrastructure.
- Install the **oc** CLI and log in.

Procedure

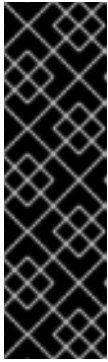
1. Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Observe the running state of your cluster.
 - a. Run the following command to view the current cluster version and status:

```
$ oc get clusterversion
```

Example output

```
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
version   False    True       24m         Working towards 4.5.4: 99% complete
```

- b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

```
$ oc get clusteroperators
```

Example output

```
NAME                               VERSION  AVAILABLE  PROGRESSING  DEGRADED  SINCE
authentication                      4.5.4   True       False        False     7m56s
cloud-credential                     4.5.4   True       False        False     31m
cluster-autoscaler                   4.5.4   True       False        False     16m
console                              4.5.4   True       False        False     10m
csi-snapshot-controller              4.5.4   True       False        False     16m
dns                                  4.5.4   True       False        False     22m
etcd                                  4.5.4   False      False        False     25s
image-registry                       4.5.4   True       False        False     16m
ingress                              4.5.4   True       False        False     16m
insights                             4.5.4   True       False        False     17m
kube-apiserver                       4.5.4   True       False        False     19m
kube-controller-manager              4.5.4   True       False        False     20m
kube-scheduler                       4.5.4   True       False        False     20m
kube-storage-version-migrator        4.5.4   True       False        False     16m
machine-api                          4.5.4   True       False        False     22m
machine-config                       4.5.4   True       False        False     22m
marketplace                          4.5.4   True       False        False     16m
monitoring                           4.5.4   True       False        False     10m
network                              4.5.4   True       False        False     23m
node-tuning                          4.5.4   True       False        False     23m
openshift-apiserver                  4.5.4   True       False        False     17m
openshift-controller-manager         4.5.4   True       False        False     15m
openshift-samples                    4.5.4   True       False        False     16m
```


operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

- c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE                               NAME
READY  STATUS  RESTARTS  AGE
kube-system                               etcd-member-ip-10-0-3-111.us-east-
2.compute.internal                       1/1    Running  0    35m
kube-system                               etcd-member-ip-10-0-3-239.us-east-
2.compute.internal                       1/1    Running  0    37m
kube-system                               etcd-member-ip-10-0-3-24.us-east-
2.compute.internal                       1/1    Running  0    35m
openshift-apiserver-operator              openshift-apiserver-operator-6d6674f4f4-
h7t2t                                     1/1    Running  1    37m
openshift-apiserver                       apiserver-fm48r
1/1    Running  0    30m
openshift-apiserver                       apiserver-fxkvv
1/1    Running  0    29m
openshift-apiserver                       apiserver-q85nm
1/1    Running  0    29m
...
openshift-service-ca-operator              openshift-service-ca-operator-66ff6dc6cd-
9r257                                     1/1    Running  0    37m
openshift-service-ca                      apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1    Running  0    35m
openshift-service-ca                      configmap-cabundle-injector-8498544d7-
25qn6                                     1/1    Running  0    35m
openshift-service-ca                      service-serving-cert-signer-6445fc9c6-wqdqn
1/1    Running  0    35m
openshift-service-catalog-apiserver-operator  openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w                1/1    Running  0    32m
openshift-service-catalog-controller-manager-operator  openshift-service-catalog-
controller-manager-operator-b78cr2lnm    1/1    Running  0    31m

```

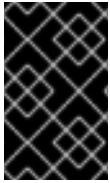
When the current cluster version is **AVAILABLE**, the installation is complete.

4.10.23. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

4.11. INSTALLING A CLUSTER ON GCP IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.5, you can install a cluster on Google Cloud Platform (GCP) that uses infrastructure that you provide and an internal mirror of the installation release content.



IMPORTANT

While you can install an OpenShift Container Platform cluster by using mirrored installation release content, your cluster still requires internet access to use the GCP APIs.

The steps for performing a user-provided infrastructure install are outlined here. Several [Deployment Manager](#) templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods.

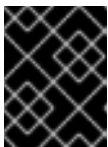


IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several Deployment Manager templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

4.11.1. Prerequisites

- [Create a registry on your mirror host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to. While you might need to grant access to more sites, you must grant access to ***.googleapis.com** and **accounts.google.com**.
- If you do not allow the system to manage identity and access management (IAM), then a cluster administrator can [manually create and maintain IAM credentials](#). Manual mode can also be used in environments where the cloud IAM APIs are not reachable.

4.11.2. Configuring your GCP project

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

4.11.2.1. Creating a GCP project

To install OpenShift Container Platform, you must create a project in your Google Cloud Platform (GCP) account to host the cluster.

Procedure

- Create a project to host your OpenShift Container Platform cluster. See [Creating and Managing Projects](#) in the GCP documentation.



IMPORTANT

Your GCP project must use the Premium Network Service Tier if you are using installer-provisioned infrastructure. The Standard Network Service Tier is not supported for clusters installed using the installation program. The installation program configures internal load balancing for the **api-int.<cluster_name>.<base_domain>** URL; the Premium Tier is required for internal load balancing.

4.11.2.2. Enabling API services in GCP

Your Google Cloud Platform (GCP) project requires access to several API services to complete OpenShift Container Platform installation.

Prerequisites

- You created a project to host your cluster.

Procedure

- Enable the following required API services in the project that hosts your cluster. See [Enabling services](#) in the GCP documentation.

Table 4.41. Required API services

API service	Console service name
Compute Engine API	compute.googleapis.com
Google Cloud APIs	cloudapis.googleapis.com
Cloud Resource Manager API	cloudresourcemanager.googleapis.com
Google DNS API	dns.googleapis.com
IAM Service Account Credentials API	iamcredentials.googleapis.com
Identity and Access Management (IAM) API	iam.googleapis.com
Service Management API	servicemanagement.googleapis.com
Service Usage API	serviceusage.googleapis.com

API service	Console service name
Google Cloud Storage JSON API	storage-api.googleapis.com
Cloud Storage	storage-component.googleapis.com

4.11.2.3. Configuring DNS for GCP

To install OpenShift Container Platform, the Google Cloud Platform (GCP) account you use must have a dedicated public hosted zone in the same project that you host the OpenShift Container Platform cluster. This zone must be authoritative for the domain. The DNS service provides cluster DNS resolution and name lookup for external connections to the cluster.

Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through GCP or another source.



NOTE

If you purchase a new domain, it can take time for the relevant DNS changes to propagate. For more information about purchasing domains through Google, see [Google Domains](#).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](#) in the GCP documentation. Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.
3. Extract the new authoritative name servers from the hosted zone records. See [Look up your Cloud DNS name servers](#) in the GCP documentation. You typically have four name servers.
4. Update the registrar records for the name servers that your domain uses. For example, if you registered your domain to Google Domains, see the following topic in the Google Domains Help: [How to switch to custom name servers](#).
5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](#) in the GCP documentation.
6. If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain. This process might include a request to your company's IT department or the division that controls the root domain and DNS services for your company.

4.11.2.4. GCP account limits

The OpenShift Container Platform cluster uses a number of Google Cloud Platform (GCP) components, but the default [Quotas](#) do not affect your ability to install a default OpenShift Container Platform cluster.

A default cluster, which contains three compute and three control plane machines, uses the following resources. Note that some resources are required only during the bootstrap process and are removed after the cluster deploys.

Table 4.42. GCP resources used in a default cluster

Service	Component	Location	Total resources required	Resources removed after bootstrap
Service account	IAM	Global	5	0
Firewall rules	Networking	Global	11	1
Forwarding rules	Compute	Global	2	0
Health checks	Compute	Global	2	0
Images	Compute	Global	1	0
Networks	Networking	Global	1	0
Routers	Networking	Global	1	0
Routes	Networking	Global	2	0
Subnetworks	Compute	Global	2	0
Target pools	Networking	Global	2	0

**NOTE**

If any of the quotas are insufficient during installation, the installation program displays an error that states both which quota was exceeded and the region.

Be sure to consider your actual cluster size, planned cluster growth, and any usage from other clusters that are associated with your account. The CPU, static IP addresses, and persistent disk SSD (storage) quotas are the ones that are most likely to be insufficient.

If you plan to deploy your cluster in one of the following regions, you will exceed the maximum storage quota and are likely to exceed the CPU quota limit:

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europa-north1**
- **europa-west2**
- **europa-west3**

- **europa-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

You can increase resource quotas from the [GCP console](#), but you might need to file a support ticket. Be sure to plan your cluster size early so that you can allow time to resolve the support ticket before you install your OpenShift Container Platform cluster.

4.11.2.5. Creating a service account in GCP

OpenShift Container Platform requires a Google Cloud Platform (GCP) service account that provides authentication and authorization to access data in the Google APIs. If you do not have an existing IAM service account that contains the required roles in your project, you must create one.

Prerequisites

- You created a project to host your cluster.

Procedure

1. Create a service account in the project that you use to host your OpenShift Container Platform cluster. See [Creating a service account](#) in the GCP documentation.
2. Grant the service account the appropriate permissions. You can either grant the individual permissions that follow or assign the **Owner** role to it. See [Granting roles to a service account for specific resources](#).



NOTE

While making the service account an owner of the project is the easiest way to gain the required permissions, it means that service account has complete control over the project. You must determine if the risk that comes from offering that power is acceptable.

3. Create the service account key in JSON format. See [Creating service account keys](#) in the GCP documentation.
The service account key is required to create a cluster.

4.11.2.5.1. Required GCP permissions

When you attach the **Owner** role to the service account that you create, you grant that service account all permissions, including those that are required to install OpenShift Container Platform. To deploy an OpenShift Container Platform cluster, the service account requires the following permissions. If you deploy your cluster into an existing VPC, the service account does not require certain networking permissions, which are noted in the following lists:

Required roles for the installation program

- Compute Admin
- Security Admin

- Service Account Admin
- Service Account User
- Storage Admin

Required roles for creating network resources during installation

- DNS Administrator

Required roles for user-provisioned GCP infrastructure

- Deployment Manager Editor
- Service Account Key Admin

Optional roles

For the cluster to create new limited credentials for its Operators, add the following role:

- Service Account Key Admin

The roles are applied to the service accounts that the control plane and compute machines use:

Table 4.43. GCP service account permissions

Account	Roles
Control Plane	roles/compute.instanceAdmin
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
Compute	roles/compute.viewer
	roles/storage.admin

4.11.2.6. Supported GCP regions

You can deploy an OpenShift Container Platform cluster to the following Google Cloud Platform (GCP) regions:

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)

- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europa-north1** (Hamina, Finland)
- **europa-west1** (St. Ghislain, Belgium)
- **europa-west2** (London, England, UK)
- **europa-west3** (Frankfurt, Germany)
- **europa-west4** (Eemshaven, Netherlands)
- **europa-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.11.2.7. Installing and configuring CLI tools for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must install and configure the CLI tools for GCP.

Prerequisites

- You created a project to host your cluster.
- You created a service account and granted it the required permissions.

Procedure

1. Install the following binaries in **\$PATH**:

- **gcloud**
- **gsutil**

See [Install the latest Cloud SDK version](#) in the GCP documentation.

2. Authenticate using the **gcloud** tool with your configured service account.
See [Authorizing with a service account](#) in the GCP documentation.

4.11.3. Creating the installation files for GCP

To install OpenShift Container Platform on Google Cloud Platform (GCP) using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files.

4.11.3.1. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

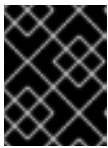


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

- iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Enter a descriptive name for your cluster.
 - viii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

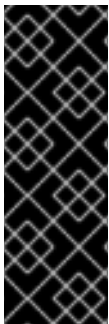


IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

4.11.3.2. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

INFO Consuming Install Config from target directory
 WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings

- 1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:

- a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.
- b. Locate the **mastersSchedulable** parameter and set its value to **False**.
- c. Save and exit the file.

5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- 1 2 Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

6. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

Additional resources

- [Optional: Adding the ingress DNS records](#)

4.11.4. Exporting common variables

4.11.4.1. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Google Cloud Platform (GCP). The provided Deployment Manager templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

4.11.4.2. Exporting common variables for Deployment Manager templates

You must export a common set of variables that are used with the provided Deployment Manager templates used to assist in completing a user-provided infrastructure install on Google Cloud Platform (GCP).



NOTE

Specific Deployment Manager templates can also require additional exported variables, which are detailed in their related procedures.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

1. Export the following common variables to be used by the provided Deployment Manager templates:

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

4.11.5. Creating a VPC in GCP

You must create a VPC in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements. One way to create the VPC is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.

Procedure

1. Copy the template from the **Deployment Manager template for the VPC** section of this topic and save it as **01_vpc.py** on your computer. This template describes the VPC that your cluster requires.
2. Create a **01_vpc.yaml** resource definition file:

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

- 1** **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2** **region** is the region to deploy the cluster into, for example **us-central1**.
- 3** **master_subnet_cidr** is the CIDR for the master subnet, for example **10.0.0.0/19**.
- 4** **worker_subnet_cidr** is the CIDR for the worker subnet, for example **10.0.32.0/19**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

4.11.5.1. Deployment Manager template for the VPC

You can use the following Deployment Manager template to deploy the VPC that you need for your OpenShift Container Platform cluster:

Example 4.19. 01_vpc.py Deployment Manager template

-

```

def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }, {
            'name': context.properties['infra_id'] + '-nat-worker',
            'natIpAllocateOption': 'AUTO_ONLY',
            'minPortsPerVm': 512,
            'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
            'subnetworks': [{
                'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                'sourceIpRangesToNat': ['ALL_IP_RANGES']
            }]
        }
    ]

    return {'resources': resources}

```

4.11.6. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 4.44. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 4.45. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 4.46. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



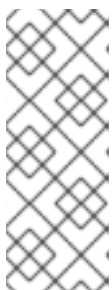
NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 4.47. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 4.48. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

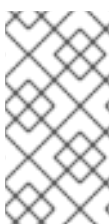


NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

4.11.7. Creating load balancers in GCP

You must configure load balancers in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the internal load balancer** section of this topic and save it as **02_lb_int.py** on your computer. This template describes the internal load balancing objects that your cluster requires.

2. For an external cluster, also copy the template from the **Deployment Manager template for the external load balancer** section of this topic and save it as **02_lb_ext.py** on your computer. This template describes the external load balancing objects that your cluster requires.
3. Export the variables that the deployment template uses:

- a. Export the cluster network location:

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
```

- b. Export the control plane subnet location:

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

- c. Export the three zones that the cluster uses:

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
```

4. Create a **02_infra.yaml** resource definition file:

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py 1
resources:
- name: cluster-lb-ext 2
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' 3
    region: '${REGION}' 4
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' 5
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: 6
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

1 **2** Required only when deploying an external cluster.

- 3 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 4 **region** is the region to deploy the cluster into, for example **us-central1**.
- 5 **control_subnet** is the URI to the control subnet.
- 6 **zones** are the zones to deploy the control plane instances into, like **us-east1-b**, **us-east1-c**, and **us-east1-d**.

5. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

4.11.7.1. Deployment Manager template for the external load balancer

You can use the following Deployment Manager template to deploy the external load balancer that you need for your OpenShift Container Platform cluster:

Example 4.20. 02_lb_ext.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$ (ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
```

```

        'instances': []
    }
}, {
    'name': context.properties['infra_id'] + '-api-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
        'region': context.properties['region'],
        'IPAddress': '${ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink}',
        'target': '${ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink}',
        'portRange': '6443'
    }
}
]]

return {'resources': resources}

```

4.11.7.2. Deployment Manager template for the internal load balancer

You can use the following Deployment Manager template to deploy the internal load balancer that you need for your OpenShift Container Platform cluster:

Example 4.21. 02_lb_int.py Deployment Manager template

```

def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '${ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink}'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
        # probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',

```

```

    'properties': {
      'backends': backends,
      'healthChecks': ['$$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
      'loadBalancingScheme': 'INTERNAL',
      'region': context.properties['region'],
      'protocol': 'TCP',
      'timeoutSec': 120
    }
  }, {
    'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
    'type': 'compute.v1.forwardingRule',
    'properties': {
      'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
      'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
      'loadBalancingScheme': 'INTERNAL',
      'ports': ['6443', '22623'],
      'region': context.properties['region'],
      'subnetwork': context.properties['control_subnet']
    }
  }
}]

for zone in context.properties['zones']:
  resources.append({
    'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
    'type': 'compute.v1.instanceGroup',
    'properties': {
      'namedPorts': [
        {
          'name': 'ignition',
          'port': 22623
        }, {
          'name': 'https',
          'port': 6443
        }
      ],
      'network': context.properties['cluster_network'],
      'zone': zone
    }
  })

return {'resources': resources}

```

You will need this template in addition to the **02_lb_ext.py** template when you create an external cluster.

4.11.8. Creating a private DNS zone in GCP

You must configure a private DNS zone in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create this component is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for the private DNS** section of this topic and save it as **02_dns.py** on your computer. This template describes the private DNS objects that your cluster requires.
2. Create a **02_dns.yaml** resource definition file:

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' ❶
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' ❷
    cluster_network: '${CLUSTER_NETWORK}' ❸
EOF
```

- ❶ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- ❷ **cluster_domain** is the domain for the cluster, for example **openshift.example.com**.
- ❸ **cluster_network** is the **selfLink** URL to the cluster network.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. The templates do not create DNS entries due to limitations of Deployment Manager, so you must create them manually:
 - a. Add the internal DNS entries:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
```

```

api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone

```

- b. For an external cluster, also add the external DNS entries:

```

$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}

```

4.11.8.1. Deployment Manager template for the private DNS

You can use the following Deployment Manager template to deploy the private DNS that you need for your OpenShift Container Platform cluster:

Example 4.22. `02_dns.py` Deployment Manager template

```

def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}

```

4.11.9. Creating firewall rules in GCP

You must create firewall rules in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for firewall rules** section of this topic and save it as **03_firewall.py** on your computer. This template describes the security groups that your cluster requires.
2. Create a **03_firewall.yaml** resource definition file:

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' ❶
    infra_id: '${INFRA_ID}' ❷
    cluster_network: '${CLUSTER_NETWORK}' ❸
    network_cidr: '${NETWORK_CIDR}' ❹
EOF
```

- ❶ **allowed_external_cidr** is the CIDR range that can access the cluster API and SSH to the bootstrap host. For an internal cluster, set this value to **\${NETWORK_CIDR}**.
- ❷ **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- ❸ **cluster_network** is the **selfLink** URL to the cluster network.
- ❹ **network_cidr** is the CIDR of the VPC network, for example **10.0.0.0/16**.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

4.11.9.1. Deployment Manager template for firewall rules

You can use the following Deployment Manager template to deploy the firewall rules that you need for your OpenShift Container Platform cluster:

Example 4.23. 03_firewall.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    ], {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
```

```

'name': context.properties['infra_id'] + '-control-plane',
'type': 'compute.v1.firewall',
'properties': {
  'network': context.properties['cluster_network'],
  'allowed': [{
    'IPProtocol': 'tcp',
    'ports': ['10257']
  },{
    'IPProtocol': 'tcp',
    'ports': ['10259']
  },{
    'IPProtocol': 'tcp',
    'ports': ['22623']
  }],
  'sourceTags': [
    context.properties['infra_id'] + '-master',
    context.properties['infra_id'] + '-worker'
  ],
  'targetTags': [context.properties['infra_id'] + '-master']
}
}, {
'name': context.properties['infra_id'] + '-internal-network',
'type': 'compute.v1.firewall',
'properties': {
  'network': context.properties['cluster_network'],
  'allowed': [{
    'IPProtocol': 'icmp'
  },{
    'IPProtocol': 'tcp',
    'ports': ['22']
  }],
  'sourceRanges': [context.properties['network_cidr']],
  'targetTags': [
    context.properties['infra_id'] + '-master',
    context.properties['infra_id'] + '-worker'
  ]
}
}, {
'name': context.properties['infra_id'] + '-internal-cluster',
'type': 'compute.v1.firewall',
'properties': {
  'network': context.properties['cluster_network'],
  'allowed': [{
    'IPProtocol': 'udp',
    'ports': ['4789', '6081']
  },{
    'IPProtocol': 'tcp',
    'ports': ['9000-9999']
  },{
    'IPProtocol': 'udp',
    'ports': ['9000-9999']
  },{
    'IPProtocol': 'tcp',
    'ports': ['10250']
  },{
    'IPProtocol': 'tcp',

```

```

      'ports': ['30000-32767']
    },{
      'IPProtocol': 'udp',
      'ports': ['30000-32767']
    }],
    'sourceTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ],
    'targetTags': [
      context.properties['infra_id'] + '-master',
      context.properties['infra_id'] + '-worker'
    ]
  }
}
}
}

return {'resources': resources}

```

4.1.10. Creating IAM roles in GCP

You must create IAM roles in Google Cloud Platform (GCP) for your OpenShift Container Platform cluster to use. One way to create these components is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your GCP infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

Procedure

1. Copy the template from the **Deployment Manager template for IAM roles** section of this topic and save it as **03_iam.py** on your computer. This template describes the IAM roles that your cluster requires.
2. Create a **03_iam.yaml** resource definition file:

```

$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py

```

```
properties:
  infra_id: '${INFRA_ID}' 1
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

3. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. Export the variable for the master service account:

```
$ export MASTER_SERVICE_ACCOUNT=$(gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '[0].email')
```

5. Export the variable for the worker service account:

```
$ export WORKER_SERVICE_ACCOUNT=$(gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email')
```

6. Export the variable for the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=$(gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink)
```

7. The templates do not create the policy bindings due to limitations of Deployment Manager, so you must create them manually:

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. Create a service account key and store it locally for later use:

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

4.11.10.1. Deployment Manager template for IAM roles

You can use the following Deployment Manager template to deploy the IAM roles that you need for your OpenShift Container Platform cluster:

Example 4.24. 03_iam.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }
    ]

    return {'resources': resources}
```

4.11.11. Creating the RHCOS cluster image for the GCP infrastructure

You must use a valid Red Hat Enterprise Linux CoreOS (RHCOS) image for Google Cloud Platform (GCP) for your OpenShift Container Platform nodes.

Procedure

1. Obtain the RHCOS image from the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The file name contains the OpenShift Container Platform version number in the format **rhcos-
<version>-<arch>-gcp.<arch>.tar.gz**.

2. Create the Google storage bucket:

```
$ gsutil mb gs://<bucket_name>
```

3. Upload the RHCOS image to the Google storage bucket:

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. Export the uploaded RHCOS image location as a variable:

```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz`
```

5. Create the cluster image:

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
--source-uri="${IMAGE_SOURCE}"
```

4.11.12. Creating the bootstrap machine in GCP

You must create the bootstrap machine in Google Cloud Platform (GCP) to use during OpenShift Container Platform cluster initialization. One way to create this machine is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your bootstrap machine, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Ensure pyOpenSSL is installed.

Procedure

1. Copy the template from the **Deployment Manager template for the bootstrap machine** section of this topic and save it as **04_bootstrap.py** on your computer. This template describes the bootstrap machine that your cluster requires.
2. Export the location of the Red Hat Enterprise Linux CoreOS (RHCOS) image that the installation program requires:

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. Create a bucket and upload the **bootstrap.ign** file:

■

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Create a signed URL for the bootstrap instance to use to access the Ignition config. Export the URL from the output as a variable:

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. Create a **04_bootstrap.yaml** resource definition file:

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2 **region** is the region to deploy the cluster into, for example **us-central1**.
- 3 **zone** is the zone to deploy the bootstrap instance into, for example **us-central1-b**.
- 4 **cluster_network** is the **selfLink** URL to the cluster network.
- 5 **control_subnet** is the **selfLink** URL to the control subnet.
- 6 **image** is the **selfLink** URL to the RHCOS image.
- 7 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 8 **root_volume_size** is the boot disk size for the bootstrap machine.
- 9 **bootstrap_ign** is the URL output when creating a signed URL.

6. Create the deployment by using the **gcloud** CLI:


```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the bootstrap machine manually.

- a. Add the bootstrap instance to the internal load balancer instance group:

```
$ gcloud compute instance-groups unmanaged add-instances \
  ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
bootstrap
```

- b. Add the bootstrap instance group to the internal load balancer backend service:

```
$ gcloud compute backend-services add-backend \
  ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
```

4.11.12.1. Deployment Manager template for the bootstrap machine

You can use the following Deployment Manager template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster:

Example 4.25. 04_bootstrap.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '"},"verification":{}}},"timeouts":{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":
{},"systemd":{}}',
                }],
            }
        }
    ]
```

```

    },
    'networkInterfaces': [{
      'subnetwork': context.properties['control_subnet'],
      'accessConfigs': [{
        'natIP': '${ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address}'
      }]
    }],
    'tags': {
      'items': [
        context.properties['infra_id'] + '-master',
        context.properties['infra_id'] + '-bootstrap'
      ]
    },
    'zone': context.properties['zone']
  }
}, {
  'name': context.properties['infra_id'] + '-bootstrap-instance-group',
  'type': 'compute.v1.instanceGroup',
  'properties': {
    'namedPorts': [
      {
        'name': 'ignition',
        'port': 22623
      }, {
        'name': 'https',
        'port': 6443
      }
    ],
    'network': context.properties['cluster_network'],
    'zone': context.properties['zone']
  }
}]

return {'resources': resources}

```

4.11.13. Creating the control plane machines in GCP

You must create the control plane machines in Google Cloud Platform (GCP) for your cluster to use. One way to create these machines is to modify the provided Deployment Manager template.



NOTE

If you do not use the provided Deployment Manager template to create your control plane machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.

Procedure

1. Copy the template from the **Deployment Manager template for control plane machines** section of this topic and save it as **05_control_plane.py** on your computer. This template describes the control plane machines that your cluster requires.
2. Export the following variable required by the resource definition:

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. Create a **05_control_plane.yaml** resource definition file:

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

- 1 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 2 **zones** are the zones to deploy the control plane instances into, for example **us-central1-a**, **us-central1-b**, and **us-central1-c**.
- 3 **control_subnet** is the **selfLink** URL to the control subnet.
- 4 **image** is the **selfLink** URL to the RHCOS image.
- 5 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 6 **service_account_email** is the email address for the master service account that you created.

7 **ignition** is the contents of the **master.ign** file.

4. Create the deployment by using the **gcloud** CLI:

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. The templates do not manage load balancer membership due to limitations of Deployment Manager, so you must add the control plane machines manually.

- Run the following commands to add the control plane machines to the appropriate instance groups:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-m-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-m-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-m-2
```

- For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

4.11.13.1. Deployment Manager template for control plane machines

You can use the following Deployment Manager template to deploy the control plane machines that you need for your OpenShift Container Platform cluster:

Example 4.26. 05_control_plane.py Deployment Manager template

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-m-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }
        ],
        'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
```

```

context.properties['machine_type'],
  'metadata': {
    'items': [{
      'key': 'user-data',
      'value': context.properties['ignition']
    }]
  },
  'networkInterfaces': [{
    'subnetwork': context.properties['control_subnet']
  }],
  'serviceAccounts': [{
    'email': context.properties['service_account_email'],
    'scopes': ['https://www.googleapis.com/auth/cloud-platform']
  }],
  'tags': {
    'items': [
      context.properties['infra_id'] + '-master',
    ]
  },
  'zone': context.properties['zones'][0]
}
}, {
  'name': context.properties['infra_id'] + '-m-1',
  'type': 'compute.v1.instance',
  'properties': {
    'disks': [{
      'autoDelete': True,
      'boot': True,
      'initializeParams': {
        'diskSizeGb': context.properties['root_volume_size'],
        'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
        'sourceImage': context.properties['image']
      }
    }],
    'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
    'metadata': {
      'items': [{
        'key': 'user-data',
        'value': context.properties['ignition']
      }]
    },
    'networkInterfaces': [{
      'subnetwork': context.properties['control_subnet']
    }],
    'serviceAccounts': [{
      'email': context.properties['service_account_email'],
      'scopes': ['https://www.googleapis.com/auth/cloud-platform']
    }],
    'tags': {
      'items': [
        context.properties['infra_id'] + '-master',
      ]
    },
    'zone': context.properties['zones'][1]
  }
}

```

```

    }, {
      'name': context.properties['infra_id'] + '-m-2',
      'type': 'compute.v1.instance',
      'properties': {
        'disks': [{
          'autoDelete': True,
          'boot': True,
          'initializeParams': {
            'diskSizeGb': context.properties['root_volume_size'],
            'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
            'sourceImage': context.properties['image']
          }
        }
      ],
      'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
      'metadata': {
        'items': [{
          'key': 'user-data',
          'value': context.properties['ignition']
        }
      ]
    },
    'networkInterfaces': [{
      'subnetwork': context.properties['control_subnet']
    }],
    'serviceAccounts': [{
      'email': context.properties['service_account_email'],
      'scopes': ['https://www.googleapis.com/auth/cloud-platform']
    }],
    'tags': {
      'items': [
        context.properties['infra_id'] + '-master',
      ]
    },
    'zone': context.properties['zones'][2]
  }
]

return {'resources': resources}

```

4.11.14. Wait for bootstrap completion and remove bootstrap resources in GCP

After you create all of the required infrastructure in Google Cloud Platform (GCP), wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.

- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Change to the directory that contains the installation program and run the following command:

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
--log-level info 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

If the command exits without a **FATAL** warning, your production control plane has initialized.

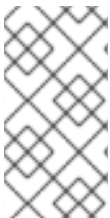
2. Delete the bootstrap resources:

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

4.11.15. Creating additional worker machines in GCP

You can create worker machines in Google Cloud Platform (GCP) for your cluster to use by launching individual instances discretely or by automated processes outside the cluster, such as auto scaling groups. You can also take advantage of the built-in cluster scaling mechanisms and the machine API in OpenShift Container Platform.

In this example, you manually launch one instance by using the Deployment Manager template. Additional instances can be launched by including additional resources of type **06_worker.py** in the file.



NOTE

If you do not use the provided Deployment Manager template to create your worker machines, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

Prerequisites

- Configure a GCP account.
- Generate the Ignition config files for your cluster.
- Create and configure a VPC and associated subnets in GCP.

- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.

Procedure

1. Copy the template from the **Deployment Manager template for worker machines** section of this topic and save it as **06_worker.py** on your computer. This template describes the worker machines that your cluster requires.
2. Export the variables that the resource definition uses.

- a. Export the subnet that hosts the compute machines:

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

- b. Export the email address for your service account:

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '[0].email`)
```

- c. Export the location of the compute machine Ignition config file:

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' 1
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 2
    zone: '${ZONE_0}' 3
    compute_subnet: '${COMPUTE_SUBNET}' 4
    image: '${CLUSTER_IMAGE}' 5
    machine_type: 'n1-standard-4' 6
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
```



```

compute_subnet: '${COMPUTE_SUBNET}' 11
image: '${CLUSTER_IMAGE}' 12
machine_type: 'n1-standard-4' 13
root_volume_size: '128'
service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
ignition: '${WORKER_IGNITION}' 15
EOF

```

- 1 **name** is the name of the worker machine, for example **worker-0**.
- 2 9 **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- 3 10 **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.
- 4 11 **compute_subnet** is the **selfLink** URL to the compute subnet.
- 5 12 **image** is the **selfLink** URL to the RHCOS image.
- 6 13 **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- 7 14 **service_account_email** is the email address for the worker service account that you created.
- 8 15 **ignition** is the contents of the **worker.ign** file.

4. Optional: If you want to launch additional instances, include additional resources of type **06_worker.py** in your **06_worker.yaml** resource definition file.
5. Create the deployment by using the **gcloud** CLI:

```

$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml

```

4.11.15.1. Deployment Manager template for worker machines

You can use the following Deployment Manager template to deploy the worker machines that you need for your OpenShift Container Platform cluster:

Example 4.27. 06_worker.py Deployment Manager template

```

def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }
        ]
    }
    ],

```

```

        'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['compute_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-worker',
            ]
        },
        'zone': context.properties['zone']
    }
}
]]

return {'resources': resources}

```

4.11.16. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.11.17. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.18.3
master-1	Ready	master	63m	v1.18.3
master-2	Ready	master	64m	v1.18.3
worker-0	NotReady	worker	76s	v1.18.3
worker-1	NotReady	worker	70s	v1.18.3

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   73m   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

4.11.18. Optional: Adding the ingress DNS records

If you removed the DNS zone configuration when creating Kubernetes manifests and generating Ignition configs, you must manually create DNS records that point at the ingress load balancer. You can create either a wildcard ***.apps.{baseDomain}**. or specific records. You can use A, CNAME, and other records per your requirements.

Prerequisites

- Configure a GCP account.
- Remove the DNS Zone configuration when creating Kubernetes manifests and generating Ignition configs.
- Create and configure a VPC and associated subnets in GCP.
- Create and configure networking and load balancers in GCP.
- Create control plane and compute roles.
- Create the bootstrap machine.
- Create the control plane machines.
- Create the worker machines.

Procedure

1. Wait for the Ingress router to create a load balancer and populate the **EXTERNAL-IP** field:

```
$ oc -n openshift-ingress get service router-default
```

Example output

```
-
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
router-default	LoadBalancer	172.30.18.154	35.233.157.184	80:32288/TCP,443:31215/TCP	98

2. Add the A record to your zones:

- To use A records:

i. Export the variable for the router IP address:

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
```

ii. Add the A record to the private zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

iii. For an external cluster, also add the A record to the public zones:

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- To add explicit domains instead of using a wildcard, create entries for each of the cluster's current routes:

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host} {"\n"}{end}{end}' routes
```

Example output

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

4.11.19. Completing a GCP installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Google Cloud Platform (GCP) user-provisioned infrastructure, you can monitor the cluster events until the cluster is ready.

Prerequisites

- Deploy the bootstrap machine for an OpenShift Container Platform cluster on user-provisioned GCP infrastructure.
- Install the **oc** CLI and log in.

Procedure

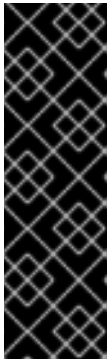
1. Complete the cluster installation:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Observe the running state of your cluster.
 - a. Run the following command to view the current cluster version and status:

```
$ oc get clusterversion
```

Example output

```
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
version   False    True       24m         Working towards 4.5.4: 99% complete
```

- b. Run the following command to view the Operators managed on the control plane by the Cluster Version Operator (CVO):

```
$ oc get clusteroperators
```

Example output

```
NAME              VERSION  AVAILABLE  PROGRESSING  DEGRADED  SINCE
authentication    4.5.4   True       False        False     7m56s
cloud-credential  4.5.4   True       False        False     31m
```

cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

- c. Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE                               NAME
READY  STATUS  RESTARTS  AGE
kube-system                               etcd-member-ip-10-0-3-111.us-east-
2.compute.internal                       1/1    Running  0    35m
kube-system                               etcd-member-ip-10-0-3-239.us-east-
2.compute.internal                       1/1    Running  0    37m
kube-system                               etcd-member-ip-10-0-3-24.us-east-
2.compute.internal                       1/1    Running  0    35m
openshift-apiserver-operator             openshift-apiserver-operator-6d6674f4f4-
h7t2t                                     1/1    Running  1    37m
openshift-apiserver                      apiserver-fm48r
1/1    Running  0    30m
openshift-apiserver                      apiserver-fxkvv
1/1    Running  0    29m
openshift-apiserver                      apiserver-q85nm
1/1    Running  0    29m
...
openshift-service-ca-operator            openshift-service-ca-operator-66ff6dc6cd-
9r257                                     1/1    Running  0    37m
openshift-service-ca                    apiservice-cabundle-injector-695b6bcbc-cl5hm

```



```

1/1    Running    0        35m
openshift-service-ca
25qn6                1/1    Running    0        35m
openshift-service-ca
1/1    Running    0        35m
openshift-service-catalog-apiserver-operator    openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w    1/1    Running    0        32m
openshift-service-catalog-controller-manager-operator    openshift-service-catalog-
controller-manager-operator-b78cr2lnm    1/1    Running    0        31m

```

When the current cluster version is **AVAILABLE**, the installation is complete.

4.11.20. Next steps

- [Customize your cluster](#).
- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to [use Operator Lifecycle Manager \(OLM\) on restricted networks](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).

4.12. UNINSTALLING A CLUSTER ON GCP

You can remove a cluster that you deployed to Google Cloud Platform (GCP).

4.12.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access. For example, some Google Cloud resources require [IAM permissions](#) in shared VPC host projects, or there might be unused [health checks that must be deleted](#).

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```

$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2

```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

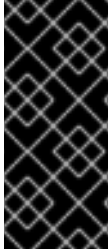
You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

CHAPTER 5. INSTALLING ON BARE METAL

5.1. INSTALLING A CLUSTER ON BARE METAL

In OpenShift Container Platform version 4.5, you can install a cluster on bare metal infrastructure that you provision.



IMPORTANT

While you might be able to follow this procedure to deploy a cluster on virtualized or cloud environments, you must be aware of additional considerations for non-bare metal platforms. Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you attempt to install an OpenShift Container Platform cluster in such an environment.

5.1.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

5.1.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

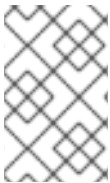
5.1.3. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

5.1.3.1. Required machines

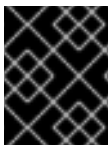
The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines. If you are running a three-node cluster, running zero compute machines is supported. Running one compute machine is not supported.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

5.1.3.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

5.1.3.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: $(\text{threads per core} \times \text{cores}) \times \text{sockets} = \text{vCPUs}$.

5.1.3.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

5.1.4. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

5.1.4.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 5.1. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 5.2. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 5.3. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 5.4. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 5.5. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources

- [Configuring chrony time service](#)

5.1.4.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 5.6. Required DNS records

Component	Record	Description
-----------	--------	-------------

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.  IMPORTANT The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 5.1. Sample DNS zone database

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 5.2. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.

```

```

98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

5.1.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```

$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1

```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

5.1.6. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.

- You need 500 MB of local disk space to download the installation program.

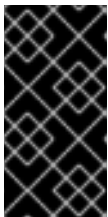
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

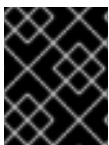
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

5.1.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

5.1.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.

3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

5.1.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**. To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

5.1.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

5.1.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

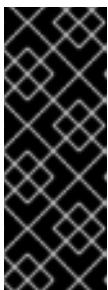
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

5.1.8.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

5.1.8.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 5.7. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object

Parameter	Description	Values
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}.{{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


5.1.8.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 5.8. Network parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>


5.1.8.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 5.9. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hypert hreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platfor m	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replica s	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

5.1.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 5.10. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

5.1.8.2. Sample install-config.yaml file for bare metal

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12

```

```
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

- 1** The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5** The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6** Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.



NOTE

Simultaneous multithreading (SMT) is enabled by default. If SMT is not enabled in your BIOS settings, the **hyperthreading** parameter has no effect.

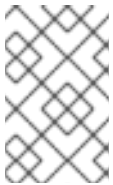


IMPORTANT

If you disable **hyperthreading**, whether in the BIOS or in the **install-config.yaml**, ensure that your capacity planning accounts for the dramatically decreased machine performance.

- 4** You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.
- 7** The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8** The cluster name that you specified in your DNS records.
- 9** A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to access the pods from an external network, you must configure load balancers and routers to manage the traffic.
- 10** The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11** The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.

- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for bare metal infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 The pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

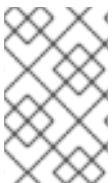


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

5.1.8.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

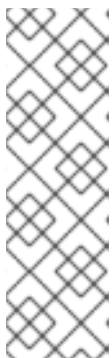


NOTE

For bare metal installations, if you do not assign node IP addresses from the range that is specified in the **networking.machineNetwork[].cidr** field in the **install-config.yaml** file, you must include them in the **proxy.noProxy** field.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

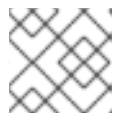
1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```

apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

5.1.9. Configuring a three-node cluster

You can optionally install and run three-node clusters in OpenShift Container Platform with no workers. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for development, production, and testing.

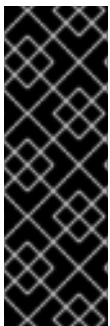
Procedure

- Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

5.1.10. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1 For `<installation_directory>`, specify the installation directory that contains the `install-config.yaml` file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.



WARNING

If you are running a three-node cluster, skip the following step to allow the masters to be schedulable.

1. Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` file.
 - b. Locate the `mastersSchedulable` parameter and set its value to `False`.
 - c. Save and exit the file.
2. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yaml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
  id: mycluster-100419-private-zone
```

```
publicZone: 2
  id: example.openshift.com
  status: {}
```

1 2 Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

3. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5.1.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. Follow either the steps to use an ISO image or network PXE booting to create the machines.

5.1.11.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.

Procedure

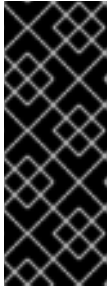
1. Upload the control plane, compute, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the [RHCOS image mirror](#) page.

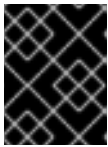


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use ISO images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

You must download the ISO file and the RAW disk file. Those file names resemble the following examples:

- ISO: **rhcos-<version>-installer.<architecture>.iso**
 - Compressed metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**
- Upload either the RAW RHCOS image file to your HTTP server and note its URL.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Use the ISO to start the RHCOS installation. Use one of the following installation options:
 - Burn the ISO image to a disk and boot it directly.
 - Use ISO redirection via a LOM interface.
- After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
- Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

- Specify the block device of the system to install to.
- Specify the URL of the RAW image that you uploaded to your server.
- Specify the URL of the Ignition config file for this machine type.
- Set **ip=dhcp** or set an individual static IP address (**ip=**) and DNS server (**nameserver=**) on each node. See *Configure advanced networking* for details.
- If you use multiple network interfaces or DNS servers, see *Configure advanced networking* for details on how to configure them.

6. Optionally, you can bond multiple network interfaces to a single interface using the **bond=** option, as described in *Configure advanced networking*.
7. Press Enter to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

5.1.11.1. Configure advanced networking

If you install Red Hat Enterprise Linux CoreOS (RHCOS) from an ISO image, you can add kernel arguments when you boot that image to configure the node's networking. The following table describes and illustrates how to use those kernel arguments.

Table 5.11. Configure advanced networking

Description	Examples
<p>To configure an IP address, either use DHCP (ip=dhcp) or set an individual static IP address (ip=<host_ip>). Then identify the DNS server IP address (nameserver=<dns_ip>) on each node. This example sets:</p> <ul style="list-style-type: none"> • The node's IP address to 10.10.10.2 • The gateway address to 10.10.10.254 • The netmask to 255.255.255.0 • The hostname to core0.example.com • The DNS server address to 4.4.4.41 	<pre>ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41</pre>
<p>Specify multiple network interfaces by specifying multiple ip= entries.</p>	<pre>ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none</pre>
<p>You can combine DHCP and static IP configurations on systems with multiple network interfaces.</p>	<pre>ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none</pre>

Description	Examples
<p>You can provide multiple DNS servers by adding a nameserver= entry for each server.</p>	<pre>nameserver=1.1.1.1 nameserver=8.8.8.8</pre>
<p>Bonding multiple network interfaces to a single interface is optionally supported using the bond= option. In these two examples:</p> <ul style="list-style-type: none"> The syntax for configuring a bonded interface is: bond=name[:network_interfaces] [:options] <i>name</i> is the bonding device name (bond0), <i>network_interfaces</i> represents a comma-separated list of physical (ethernet) interfaces (em1,em2), and <i>options</i> is a comma-separated list of bonding options. (Enter modinfo bonding to see available options.) When you create a bonded interface using bond=, you must specify how the IP address is assigned and other information for the bonded interface. 	<p>To configure the bonded interface to use DHCP, set the bond's IP address to dhcp. For example:</p> <pre>bond=bond0:em1,em2:mode=active-backup ip=bond0:dhcp</pre> <p>To configure the bonded interface to use a static IP address, enter the specific IP address you want and related information. For example:</p> <pre>bond=bond0:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:co re0.example.com:bond0:none</pre> <div data-bbox="815 987 922 1397" style="background-color: black; width: 67px; height: 183px; margin-bottom: 10px;"></div> <p>IMPORTANT</p> <p>When using the advanced networking options, you could encounter issues during the first boot of RHCOS where the statically configured address is not present or not activated properly. In such cases, you might need to manually reboot the RHCOS machine to workaround this problem. In newer versions of RHCOS, this issue is resolved. See BZ#1902584 for additional details.</p>

5.1.11.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

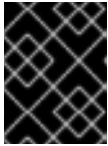
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Familiarity configuring the necessary DHCP, TFTP, and HTTP services for providing PXE or iPXE infrastructure.
- Have access to an HTTP server and TFTP server that you can access from your computer.

Procedure

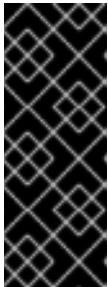
1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the compressed metal RAW image, **kernel** and **initramfs** files from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.

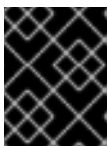


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use RAW images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- Compressed metal RAW image: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
 - **kernel**: **rhcos-<version>-<architecture>-installer-kernel-<architecture>**
 - **initramfs**: **rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**
3. Upload the RAW image to your HTTP server.
 4. Upload the additional files that are required for your booting method:
 - For traditional PXE, upload the **kernel** and **initramfs** files to your TFTP server.
 - For iPXE, upload the **kernel** and **initramfs** files to your HTTP server.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
6. Configure PXE or iPXE installation for the RHCOS images. Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```
DEFAULT pxeboot
TIMEOUT 20
```

```
PROMPT 0
```

```
LABEL pxeboot
```

```
  KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> 1
```

```
  APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-  
  initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda  
  coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.  
  <architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2
```

```
3
```

- 1** Specify the location of the **kernel** file available on your TFTP server.
- 2** If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3** Specify locations of the RHCOS files that you uploaded to your HTTP or TFTP server. The **initrd** parameter value is the location of the **initramfs** file on your TFTP server. The **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image on your HTTP server, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file on your HTTP server.



NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

- For iPXE:

```
kernel http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel-  
<architecture> ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-  
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda  
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.  
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
```

```
2
```

```
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.  
<architecture>.img 3
```

```
boot
```

- 1** Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value references the name of the **initramfs** file that is supplied on the following **initrd** line, the **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.
- 2** If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3** Specify the location of the **initramfs** file that you uploaded to your HTTP server.

**NOTE**

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **kernel** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

7. If you use UEFI, perform the following actions:

- a. Provide the EFI binaries and **grub.cfg** file that are required for booting the system. You need the **shim.efi** binary and the **grubx64.efi** binary.
 - Extract the necessary EFI binaries by mounting the RHCOS ISO on your host and then mounting the **images/efiboot.img** file to your host. From the **efiboot.img** mount point, you then copy the **EFI/redhat/shimx64.efi** and **EFI/redhat/grubx64.efi** files to your TFTP server.

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

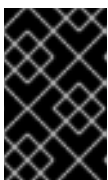
- b. Copy the **EFI/redhat/grub.cfg** file that is included in the RHCOS ISO to your TFTP server.
- c. Edit the **grub.cfg** file to include the following arguments:

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
  linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
  rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
  coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
  <architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
  initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

1 The first argument to the **linux** line item is the location of the **kernel** file that you uploaded to your TFTP server. For the **coreos.inst.image_url** parameter value, specify the location of the compressed metal RAW image that you uploaded to your HTTP server. For the **coreos.inst.ignition_url** parameter, specify the location of the bootstrap Ignition config file that you uploaded to your HTTP server.

2 Specify the location of the **initramfs** file that you uploaded to your TFTP server.

8. Continue to create the machines for your cluster.

**IMPORTANT**

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

5.1.12. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

5.1.13. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

5.1.14. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

■

Example output

```

NAME      AGE      REQUESTOR                                     CONDITION
csr-8b2br 15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE      REQUESTOR                                     CONDITION
csr-bfd72 5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...

```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}\n' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

5.1.15. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

5.1.15.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

5.1.15.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

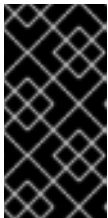
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

5.1.15.2.1. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

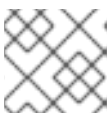


NOTE

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

5.1.15.2.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

5.1.15.2.3. Configuring block registry storage for bare metal

To allow the image registry to use block storage types during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only one (**1**) replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
3. Edit the registry configuration so that it references the correct PVC.

5.1.16. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m

insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE          NAME          READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running    1    9m
openshift-apiserver          apiserver-67b9g          1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx          1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4          1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8  1/1
Running    0    5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> 1
```

- 1** Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

5.1.17. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- [Set up your registry and configure registry storage](#) .

5.2. INSTALLING A CLUSTER ON BARE METAL WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster on bare metal infrastructure that you provision with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

5.2.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

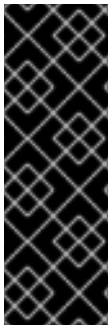
5.2.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

5.2.3. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

5.2.3.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines. If you are running a three-node cluster, running zero compute machines is supported. Running one compute machine is not supported.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

5.2.3.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

5.2.3.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: $(\text{threads per core} \times \text{cores}) \times \text{sockets} = \text{vCPUs}$.

5.2.3.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

5.2.4. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

5.2.4.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 5.12. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 5.13. All machines to control plane

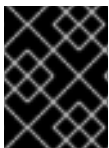
Protocol	Port	Description
TCP	6443	Kubernetes API

Table 5.14. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 5.15. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server

Port	Back-end machines (pool members)	Internal	External	Description
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the `/readyz` endpoint to the removal of the API server instance from the pool. Within the time frame after `/readyz` returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

- Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 5.16. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources

- [Configuring chrony time service](#)

5.2.4.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 5.17. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.
		 <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.</p>
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.

Component	Record	Description
Master hosts	<master><n>. <cluster_name>. <base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>. <cluster_name>. <base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 5.3. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
```

```

;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 5.4. Sample DNS zone database for reverse records

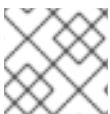
```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

5.2.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

5.2.6. Obtaining the installation program

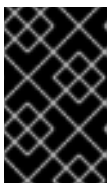
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

5.2.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

5.2.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

5.2.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

-


```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

5.2.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

5.2.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

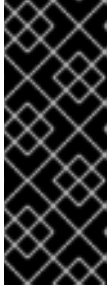
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

**IMPORTANT**

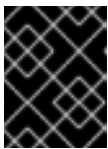
You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

**NOTE**

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

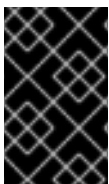
The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

5.2.8.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

5.2.8.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 5.18. Required parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object


Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


5.2.8.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 5.19. Network parameters


Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


5.2.8.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 5.20. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;"> <div style="background-color: black; width: 60px; height: 100px; margin-right: 10px;">  </div> <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}

Parameter	Description	Values
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <p> NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

5.2.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 5.21. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

5.2.8.2. Sample install-config.yaml file for bare metal

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15

```

1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.

2 5 The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

3 6 Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.



NOTE

Simultaneous multithreading (SMT) is enabled by default. If SMT is not enabled in your BIOS settings, the **hyperthreading** parameter has no effect.



IMPORTANT

If you disable **hyperthreading**, whether in the BIOS or in the **install-config.yaml**, ensure that your capacity planning accounts for the dramatically decreased machine performance.

You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not

- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8 The cluster name that you specified in your DNS records.
- 9 A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to access the pods from an external network, you must configure load balancers and routers to manage the traffic.
- 10 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for bare metal infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 The pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

5.2.8.3. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 5.22. Required network parameters

Parameter	Description	Value
networking.net workType	The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5.	The default value is OpenShiftSDN .
networking.clus terNetwork[].cid r	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .
networking.clus terNetwork[].ho stPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serv iceNetwork[]	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.mac hineNetwork[].ci dr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

5.2.9. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster. Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of [kube-proxy](#) settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.
- Create the Ignition config files for your cluster.

Procedure

1. Use the following command to create manifests:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1 For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, several network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

- 1 The parameters for the **spec** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.
5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

5.2.10. Cluster Network Operator configuration

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: ❶
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: ❷
  - 172.30.0.0/16
  defaultNetwork: ❸
  ...
  kubeProxyConfig: ❹
  iptablesSyncPeriod: 30s ❺
  proxyArguments:
    iptables-min-sync-period: ❻
    - 0s
```

- ❶ ❷ Specified in the **install-config.yaml** file.
- ❸ Configures the default Container Network Interface (CNI) network provider for the cluster network.
- ❹ The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.
- ❺ The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.



NOTE

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

- 6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time](#)

5.2.10.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
  mtu: 1450 4
  vxlanPort: 4789 5
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**.

- 5 The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for VXLAN, since both SDNs use the same default VXLAN port number.

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

5.2.10.2. Cluster Network Operator example configuration

A complete CR object for the CNO is displayed in the following example:

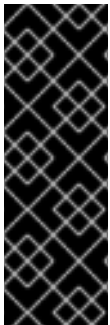
Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
```

```
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 0s
```

5.2.11. Creating the Ignition config files

Because you must manually start the cluster machines, you must generate the Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

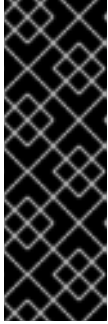
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

- Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

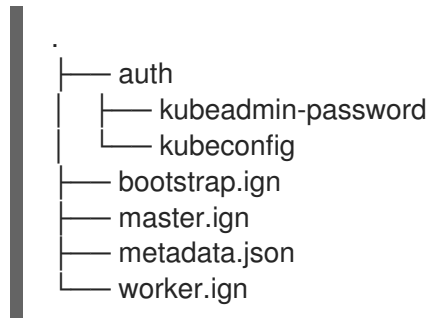
- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

If you created an **install-config.yaml** file, specify the directory that contains it. Otherwise, specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

The following files are generated in the directory:



5.2.12. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. Follow either the steps to use an ISO image or network PXE booting to create the machines.

5.2.12.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

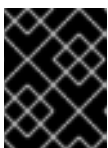
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.

Procedure

1. Upload the control plane, compute, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the [RHCOS image mirror](#) page.

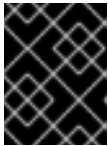


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use ISO images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

You must download the ISO file and the RAW disk file. Those file names resemble the following examples:

- ISO: **rhcos-<version>-installer.<architecture>.iso**
 - Compressed metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**
3. Upload either the RAW RHCOS image file to your HTTP server and note its URL.



IMPORTANT

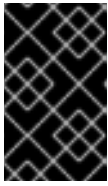
If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

4. Use the ISO to start the RHCOS installation. Use one of the following installation options:
- Burn the ISO image to a disk and boot it directly.
 - Use ISO redirection via a LOM interface.
5. After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
6. Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

- 1 Specify the block device of the system to install to.
- 2 Specify the URL of the RAW image that you uploaded to your server.
- 3 Specify the URL of the Ignition config file for this machine type.
- 4 Set **ip=dhcp** or set an individual static IP address (**ip=**) and DNS server (**nameserver=**) on each node. See *Configure advanced networking* for details.
- 5 If you use multiple network interfaces or DNS servers, see *Configure advanced networking* for details on how to configure them.
- 6 Optionally, you can bond multiple network interfaces to a single interface using the **bond=** option, as described in *Configure advanced networking*.

7. Press Enter to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
8. Continue to create the machines for your cluster.



IMPORTANT


You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

5.2.12.1.1. Configure advanced networking

If you install Red Hat Enterprise Linux CoreOS (RHCOS) from an ISO image, you can add kernel arguments when you boot that image to configure the node's networking. The following table describes and illustrates how to use those kernel arguments.

Table 5.23. Configure advanced networking

Description	Examples
<p>To configure an IP address, either use DHCP (ip=dhcp) or set an individual static IP address (ip=<host_ip>). Then identify the DNS server IP address (nameserver=<dns_ip>) on each node. This example sets:</p> <ul style="list-style-type: none"> ● The node's IP address to 10.10.10.2 ● The gateway address to 10.10.10.254 ● The netmask to 255.255.255.0 ● The hostname to core0.example.com ● The DNS server address to 4.4.4.41 	<pre>ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41</pre>
<p>Specify multiple network interfaces by specifying multiple ip= entries.</p>	<pre>ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none</pre>
<p>You can combine DHCP and static IP configurations on systems with multiple network interfaces.</p>	<pre>ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none</pre>
<p>You can provide multiple DNS servers by adding a nameserver= entry for each server.</p>	<pre>nameserver=1.1.1.1 nameserver=8.8.8.8</pre>

Description	Examples
<p>Bonding multiple network interfaces to a single interface is optionally supported using the bond= option. In these two examples:</p> <ul style="list-style-type: none"> • The syntax for configuring a bonded interface is: bond=name[:network_interfaces] [:options] • <i>name</i> is the bonding device name (bond0), <i>network_interfaces</i> represents a comma-separated list of physical (ethernet) interfaces (em1,em2), and <i>options</i> is a comma-separated list of bonding options. (Enter modinfo bonding to see available options.) • When you create a bonded interface using bond=, you must specify how the IP address is assigned and other information for the bonded interface. 	<p>To configure the bonded interface to use DHCP, set the bond's IP address to dhcp. For example:</p> <pre data-bbox="815 315 1417 405">bond=bond0:em1,em2:mode=active-backup ip=bond0:dhcp</pre> <p>To configure the bonded interface to use a static IP address, enter the specific IP address you want and related information. For example:</p> <pre data-bbox="815 584 1444 696">bond=bond0:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:co re0.example.com:bond0:none</pre> <div data-bbox="815 757 922 1167" style="background-color: black; color: white; padding: 10px; text-align: center;">  </div> <p>IMPORTANT</p> <p>When using the advanced networking options, you could encounter issues during the first boot of RHCOS where the statically configured address is not present or not activated properly. In such cases, you might need to manually reboot the RHCOS machine to workaround this problem. In newer versions of RHCOS, this issue is resolved. See BZ#1902584 for additional details.</p>

5.2.12.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

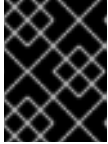
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Familiarity configuring the necessary DHCP, TFTP, and HTTP services for providing PXE or iPXE infrastructure.
- Have access to an HTTP server and TFTP server that you can access from your computer.

Procedure

1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Obtain the compressed metal RAW image, **kernel** and **initramfs** files from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.

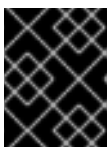


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use RAW images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- Compressed metal RAW image: **rhcos-`<version>`-`<architecture>`-metal.`<architecture>`.raw.gz**
 - kernel:** **rhcos-`<version>`-`<architecture>`-installer-kernel-`<architecture>`**
 - initramfs:** **rhcos-`<version>`-`<architecture>`-installer-initramfs.`<architecture>`.img**
- Upload the RAW image to your HTTP server.
 - Upload the additional files that are required for your booting method:
 - For traditional PXE, upload the **kernel** and **initramfs** files to your TFTP server.
 - For iPXE, upload the **kernel** and **initramfs** files to your HTTP server.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
- Configure PXE or iPXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```

DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
  KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> 1
  APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-

```

```
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
```

3

- 1 Specify the location of the **kernel** file available on your TFTP server.
- 2 If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3 Specify locations of the RHCOS files that you uploaded to your HTTP or TFTP server. The **initrd** parameter value is the location of the **initramfs** file on your TFTP server. The **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image on your HTTP server, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file on your HTTP server.



NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

- For iPXE:

```
kernel http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel-
<architecture> ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
```

2

```
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.
<architecture>.img
```

3

```
boot
```

- 1 Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value references the name of the **initramfs** file that is supplied on the following **initrd** line, the **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.
- 2 If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3 Specify the location of the **initramfs** file that you uploaded to your HTTP server.

**NOTE**

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **kernel** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

7. If you use UEFI, perform the following actions:

- a. Provide the EFI binaries and **grub.cfg** file that are required for booting the system. You need the **shim.efi** binary and the **grubx64.efi** binary.
 - Extract the necessary EFI binaries by mounting the RHCOS ISO on your host and then mounting the **images/efiboot.img** file to your host. From the **efiboot.img** mount point, you then copy the **EFI/redhat/shimx64.efi** and **EFI/redhat/grubx64.efi** files to your TFTP server.

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

- b. Copy the **EFI/redhat/grub.cfg** file that is included in the RHCOS ISO to your TFTP server.
- c. Edit the **grub.cfg** file to include the following arguments:

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
  linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
  rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
  coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
  <architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
  initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

- 1** The first argument to the **linux** line item is the location of the **kernel** file that you uploaded to your TFTP server. For the **coreos.inst.image_url** parameter value, specify the location of the compressed metal RAW image that you uploaded to your HTTP server. For the **coreos.inst.ignition_url** parameter, specify the location of the bootstrap Ignition config file that you uploaded to your HTTP server.

- 2** Specify the location of the **initramfs** file that you uploaded to your TFTP server.

8. Continue to create the machines for your cluster.

**IMPORTANT**

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

5.2.13. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1  
--log-level=info 2
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...  
INFO API v1.18.3 up  
INFO Waiting up to 30m0s for bootstrapping to complete...  
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

5.2.14. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

5.2.15. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

■

Example output

```

NAME      AGE      REQUESTOR                                     CONDITION
csr-8b2br 15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE      REQUESTOR                                     CONDITION
csr-bfd72 5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...

```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

5.2.16. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

5.2.16.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

5.2.16.2. Image registry storage configuration

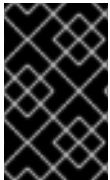
The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

5.2.16.3. Configuring block registry storage for bare metal

To allow the image registry to use block storage types during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only one (**1**) replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
3. Edit the registry configuration so that it references the correct PVC.

5.2.17. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.5.4	True	False	False 7m56s

cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE          NAME                                     READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running    1    9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0    5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> ❶
```

- ❶ Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

5.2.18. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- [Set up your registry and configure registry storage](#) .

5.3. INSTALLING A CLUSTER ON BARE METAL IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.5, you can install a cluster on bare metal infrastructure that you provision in a restricted network.



IMPORTANT

While you might be able to follow this procedure to deploy a cluster on virtualized or cloud environments, you must be aware of additional considerations for non-bare metal platforms. Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you attempt to install an OpenShift Container Platform cluster in such an environment.

5.3.1. Prerequisites

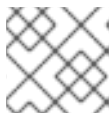
- [Create a registry on your mirror host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide ReadWriteMany access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

5.3.2. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

5.3.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

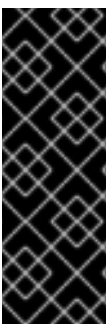
5.3.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

5.3.4. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

5.3.4.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines. If you are running a three-node cluster, running zero compute machines is supported. Running one compute machine is not supported.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#).

5.3.4.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

5.3.4.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: $(\text{threads per core} \times \text{cores}) \times \text{sockets} = \text{vCPUs}$.

5.3.4.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

5.3.5. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

5.3.5.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 5.24. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .

Protocol	Port	Description
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 5.25. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 5.26. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

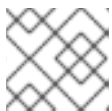
Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 5.27. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer:** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 5.28. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources


- [Configuring chrony time service](#)

5.3.5.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 5.29. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.
		 <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.</p>

Component	Record	Description
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 5.5. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H ; refresh (3 hours)
  30M ; retry (30 minutes)
  2W ; expiry (2 weeks)
  1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
```

```

; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 5.6. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

5.3.6. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

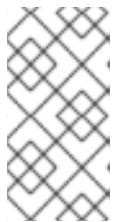
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

■

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

5.3.7. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.
- Obtain the **imageContentSources** section from the output of the command to mirror the repository.
- Obtain the contents of the certificate for your mirror registry.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

**IMPORTANT**

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

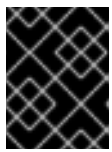
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

**NOTE**

You must name this configuration file **install-config.yaml**.

- Unless you use a registry that RHCOS trusts by default, such as **docker.io**, you must provide the contents of the certificate for your mirror repository in the **additionalTrustBundle** section. In most cases, you must provide the certificate for your mirror.
- You must include the **imageContentSources** section from the output of the command to mirror the repository.

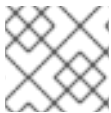
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

5.3.7.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

5.3.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 5.30. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String


Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


5.3.7.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 5.31. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .


Parameter	Description	Values
networking.serviceNetwork	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	<p>Required if you use networking.machineNetwork. An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24.</p>	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div>


5.3.7.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:

Table 5.32. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .

Parameter	Description	Values
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

5.3.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 5.33. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

5.3.7.2. Sample install-config.yaml file for bare metal

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
```

```

- hyperthreading: Enabled ③
  name: worker
  replicas: 0 ④
controlPlane: ⑤
  hyperthreading: Enabled ⑥
  name: master
  replicas: 3 ⑦
metadata:
  name: test ⑧
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 ⑨
    hostPrefix: 23 ⑩
  networkType: OpenShiftSDN
  serviceNetwork: ⑪
  - 172.30.0.0/16
platform:
  none: {} ⑫
fips: false ⑬
pullSecret: '{"auths":{"<local_registry>":{"auth": "<credentials>","email": "you@example.com"}}}' ⑭
sshKey: 'ssh-ed25519 AAAA...' ⑮
additionalTrustBundle: | ⑯
-----BEGIN CERTIFICATE-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END CERTIFICATE-----
imageContentSources: ⑰
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release

```

- ① The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- ② ⑤ The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- ③ ⑥ Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.



NOTE

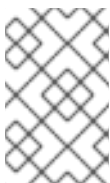
Simultaneous multithreading (SMT) is enabled by default. If SMT is not enabled in your BIOS settings, the **hyperthreading** parameter has no effect.



IMPORTANT

If you disable **hyperthreading**, whether in the BIOS or in the **install-config.yaml**, ensure that your capacity planning accounts for the dramatically decreased machine performance.

- 4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.
- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8 The cluster name that you specified in your DNS records.
- 9 A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to access the pods from an external network, you must configure load balancers and routers to manage the traffic.
- 10 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for bare metal infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 16 Provide the contents of the certificate file that you used for your mirror registry.
- 17 Provide the **imageContentSources** section from the output of the command to mirror the

5.3.7.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

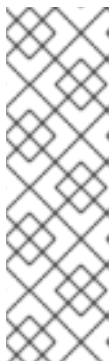


NOTE

For bare metal installations, if you do not assign node IP addresses from the range that is specified in the **networking.machineNetwork[].cidr** field in the **install-config.yaml** file, you must include them in the **proxy.noProxy** field.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.

2

A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an

- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with `.` to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use `*` to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.



NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

5.3.8. Configuring a three-node cluster

You can optionally install and run three-node clusters in OpenShift Container Platform with no workers. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for development, production, and testing.

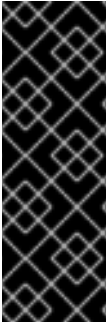
Procedure

- Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

5.3.9. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

**WARNING**

If you are running a three-node cluster, skip the following step to allow the masters to be schedulable.

1. Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
 - b. Locate the `mastersSchedulable` parameter and set its value to **False**.
 - c. Save and exit the file.
2. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}
```

- ❶ ❷ Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

3. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> ❶
```

- ❶ For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
└── bootstrap.ign
```



```

├── master.ign
├── metadata.json
└── worker.ign

```

5.3.10. Configuring chrony time service

You must set the time server and related settings used by the chrony time service (**chronyd**) by modifying the contents of the **chrony.conf** file and passing those contents to your nodes as a machine config.

Procedure

1. Create the contents of the **chrony.conf** file and encode it as base64. For example:

```

$ cat << EOF | base64
pool 0.rhel.pool.ntp.org iburst 1
driftfile /var/lib/chrony/drift
makestep 1.0 3
rtcsync
logdir /var/log/chrony
EOF

```

- 1 Specify any valid, reachable time source, such as the one provided by your DHCP server.

Example output

```

ICAgIHNIcnZlciBjbG9jay5yZWROyYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXlvaGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2
RpciAv
dmFyL2xvZy9jaHJvbnkK

```

2. Create the **MachineConfig** object file, replacing the base64 string with the one you just created. This example adds the file to **master** nodes. You can change it to **worker** or make an additional MachineConfig for the **worker** role. Create MachineConfig files for each type of machine that your cluster uses:

```

$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}

```

```

storage:
  files:
  - contents:
      source: data:text/plain;charset=utf-
8;base64,ICAgIHNIcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92Y
XlIvbGliL2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAg
IGxvZ2RpciAvdmFyL2xvZy9jaHJvbnkK
      verification: {}
      filesystem: root
      mode: 420
      path: /etc/chrony.conf
      osImageURL: ""
EOF

```

3. Make a backup copy of the configuration files.
4. Apply the configurations in one of two ways:
 - If the cluster is not up yet, after you generate manifest files, add this file to the **<installation_directory>/openshift** directory, and then continue to create the cluster.
 - If the cluster is already running, apply the file:

```
$ oc apply -f ./99-masters-chrony-configuration.yaml
```

5.3.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. Follow either the steps to use an ISO image or network PXE booting to create the machines.

5.3.11.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.

Procedure

1. Upload the control plane, compute, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use ISO images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

You must download the ISO file and the RAW disk file. Those file names resemble the following examples:

- ISO: **rhcos-`<version>`-installer.`<architecture>`.iso**
- Compressed metal RAW: **rhcos-`<version>`-metal.`<architecture>`.raw.gz**

- Upload either the RAW RHCOS image file to your HTTP server and note its URL.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Use the ISO to start the RHCOS installation. Use one of the following installation options:
 - Burn the ISO image to a disk and boot it directly.
 - Use ISO redirection via a LOM interface.
- After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
- Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

- Specify the block device of the system to install to.
- Specify the URL of the RAW image that you uploaded to your server.
- Specify the URL of the Ignition config file for this machine type.
- Set **ip=dhcp** or set an individual static IP address (**ip=**) and DNS server (**nameserver=**) on each node. See *Configure advanced networking* for details.
- If you use multiple network interfaces or DNS servers, see *Configure advanced networking* for details on how to configure them.

6. Optionally, you can bond multiple network interfaces to a single interface using the **bond=** option, as described in *Configure advanced networking*.

7. Press Enter to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

5.3.11.1.1. Configure advanced networking

If you install Red Hat Enterprise Linux CoreOS (RHCOS) from an ISO image, you can add kernel arguments when you boot that image to configure the node's networking. The following table describes and illustrates how to use those kernel arguments.

Table 5.34. Configure advanced networking

Description	Examples
<p>To configure an IP address, either use DHCP (ip=dhcp) or set an individual static IP address (ip=<host_ip>). Then identify the DNS server IP address (nameserver=<dns_ip>) on each node. This example sets:</p> <ul style="list-style-type: none"> • The node's IP address to 10.10.10.2 • The gateway address to 10.10.10.254 • The netmask to 255.255.255.0 • The hostname to core0.example.com • The DNS server address to 4.4.4.41 	<pre>ip=10.10.10.2::10.10.10.254:255.255.255.0:co re0.example.com:enp1s0:none nameserver=4.4.4.41</pre>
<p>Specify multiple network interfaces by specifying multiple ip= entries.</p>	<pre>ip=10.10.10.2::10.10.10.254:255.255.255.0:co re0.example.com:enp1s0:none ip=10.10.10.3::10.10.10.254:255.255.255.0:co re0.example.com:enp2s0:none</pre>
<p>You can combine DHCP and static IP configurations on systems with multiple network interfaces.</p>	<pre>ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:co re0.example.com:enp2s0:none</pre>

Description	Examples
<p>You can provide multiple DNS servers by adding a nameserver= entry for each server.</p>	<pre>nameserver=1.1.1.1 nameserver=8.8.8.8</pre>
<p>Bonding multiple network interfaces to a single interface is optionally supported using the bond= option. In these two examples:</p> <ul style="list-style-type: none"> The syntax for configuring a bonded interface is: bond=name[:network_interfaces] [:options] <i>name</i> is the bonding device name (bond0), <i>network_interfaces</i> represents a comma-separated list of physical (ethernet) interfaces (em1,em2), and <i>options</i> is a comma-separated list of bonding options. (Enter modinfo bonding to see available options.) When you create a bonded interface using bond=, you must specify how the IP address is assigned and other information for the bonded interface. 	<p>To configure the bonded interface to use DHCP, set the bond's IP address to dhcp. For example:</p> <pre>bond=bond0:em1,em2:mode=active-backup ip=bond0:dhcp</pre> <p>To configure the bonded interface to use a static IP address, enter the specific IP address you want and related information. For example:</p> <pre>bond=bond0:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:co re0.example.com:bond0:none</pre> <div data-bbox="815 936 922 1346" style="background-color: black; color: white; padding: 10px; margin: 10px 0;"> <p>IMPORTANT</p> <p>When using the advanced networking options, you could encounter issues during the first boot of RHCOS where the statically configured address is not present or not activated properly. In such cases, you might need to manually reboot the RHCOS machine to workaround this problem. In newer versions of RHCOS, this issue is resolved. See BZ#1902584 for additional details.</p> </div>

5.3.11.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

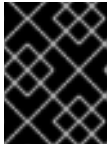
Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Familiarity configuring the necessary DHCP, TFTP, and HTTP services for providing PXE or iPXE infrastructure.
- Have access to an HTTP server and TFTP server that you can access from your computer.

Procedure

1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the compressed metal RAW image, **kernel** and **initramfs** files from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use RAW images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- Compressed metal RAW image: **rhcos-`<version>`-`<architecture>`-metal.`<architecture>`.raw.gz**
 - **kernel:** **rhcos-`<version>`-`<architecture>`-installer-kernel-`<architecture>`**
 - **initramfs:** **rhcos-`<version>`-`<architecture>`-installer-initramfs.`<architecture>`.img**
3. Upload the RAW image to your HTTP server.
 4. Upload the additional files that are required for your booting method:
 - For traditional PXE, upload the **kernel** and **initramfs** files to your TFTP server.
 - For iPXE, upload the **kernel** and **initramfs** files to your HTTP server.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
6. Configure PXE or iPXE installation for the RHCOS images. Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```

DEFAULT pxeboot
TIMEOUT 20

```

PROMPT 0

LABEL pxeboot

KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> **1**

APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.

<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign **2**

3

- 1** Specify the location of the **kernel** file available on your TFTP server.
- 2** If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3** Specify locations of the RHCOS files that you uploaded to your HTTP or TFTP server. The **initrd** parameter value is the location of the **initramfs** file on your TFTP server. The **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image on your HTTP server, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file on your HTTP server.



NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

- For iPXE:

```
kernel http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel-
<architecture> ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
```

2

```
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.
<architecture>.img 3
```

boot

- 1** Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value references the name of the **initramfs** file that is supplied on the following **initrd** line, the **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.
- 2** If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3** Specify the location of the **initramfs** file that you uploaded to your HTTP server.

**NOTE**

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **kernel** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

7. If you use UEFI, perform the following actions:

- a. Provide the EFI binaries and **grub.cfg** file that are required for booting the system. You need the **shim.efi** binary and the **grubx64.efi** binary.
 - Extract the necessary EFI binaries by mounting the RHCOS ISO on your host and then mounting the **images/efiboot.img** file to your host. From the **efiboot.img** mount point, you then copy the **EFI/redhat/shimx64.efi** and **EFI/redhat/grubx64.efi** files to your TFTP server.

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

- b. Copy the **EFI/redhat/grub.cfg** file that is included in the RHCOS ISO to your TFTP server.
- c. Edit the **grub.cfg** file to include the following arguments:

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
  linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
  rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
  coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
  <architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
  initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

- 1** The first argument to the **linux** line item is the location of the **kernel** file that you uploaded to your TFTP server. For the **coreos.inst.image_url** parameter value, specify the location of the compressed metal RAW image that you uploaded to your HTTP server. For the **coreos.inst.ignition_url** parameter, specify the location of the bootstrap Ignition config file that you uploaded to your HTTP server.

- 2** Specify the location of the **initramfs** file that you uploaded to your TFTP server.

8. Continue to create the machines for your cluster.

**IMPORTANT**

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

5.3.12. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

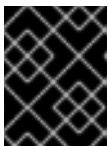
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

5.3.13. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.

- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

5.3.14. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready     master   63m   v1.18.3
master-1  Ready     master   63m   v1.18.3
master-2  Ready     master   64m   v1.18.3
worker-0  NotReady  worker   76s   v1.18.3
worker-1  NotReady  worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE   REQUESTOR                                CONDITION
csr-8b2br 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> ❶
```

- ❶** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE   REQUESTOR                                CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...

```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** `<csr_name>` is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

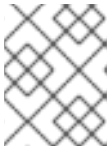
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

5.3.15. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

- Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

5.3.15.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

5.3.15.1.1. Changing the image registry's management state

To start the image registry, you must change the Image Registry Operator configuration's **managementState** from **Removed** to **Managed**.

Procedure

- Change **managementState** Image Registry Operator configuration from **Removed** to **Managed**. For example:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"managementState": "Managed"}}'
```

5.3.15.1.2. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.



NOTE

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

5.3.15.1.3. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

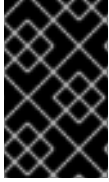
```
managementState: Removed
```

to

```
managementState: Managed
```

5.3.15.1.4. Configuring block registry storage for bare metal

To allow the image registry to use block storage types during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only one (**1**) replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
3. Edit the registry configuration so that it references the correct PVC.

5.3.16. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m

machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

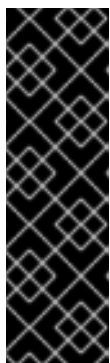
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE          NAME          READY STATUS
```

```

RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running 1 9m
openshift-apiserver  apiserver-67b9g  1/1  Running  0
3m
openshift-apiserver  apiserver-ljcmx  1/1  Running  0
1m
openshift-apiserver  apiserver-z25h4  1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8  1/1
Running 0 5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> ❶
```

- ❶ Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

3. Register your cluster on the [Cluster registration](#) page.

5.3.17. Next steps

- [Customize your cluster](#).
- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to [use Operator Lifecycle Manager \(OLM\) on restricted networks](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).

CHAPTER 6. INSTALLING ON IBM Z AND LINUXONE

6.1. INSTALLING A CLUSTER ON IBM Z AND LINUXONE

In OpenShift Container Platform version 4.5, you can install a cluster on IBM Z or LinuxONE infrastructure that you provision.



NOTE

While this document refers only to IBM Z, all information in it also applies to LinuxONE.



IMPORTANT

Additional considerations exist for non-bare metal platforms. Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you install an OpenShift Container Platform cluster.

6.1.1. Prerequisites

- Before you begin the installation process, you must move or remove any existing installation files. This ensures that the required installation files are created and updated during the installation process.
- Provision [persistent storage using NFS](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

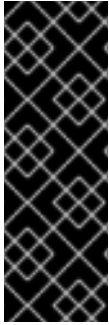
6.1.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

6.1.3. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

6.1.3.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To improve high availability of your cluster, distribute the control plane machines over different z/VM instances on at least two physical machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

6.1.3.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. The machines are configured with static IP addresses. No DHCP server is required. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server.

6.1.3.3. IBM Z network connectivity requirements

To install on IBM Z under z/VM, you require a single z/VM virtual NIC in layer 2 mode. You also need:

- A direct-attached OSA or RoCE network adapter

- A z/VM VSWITCH set up. For a preferred setup, use OSA link aggregation.

6.1.3.4. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS	2	8 GB	120 GB

1. 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

6.1.3.5. Minimum IBM Z system requirements

You can install OpenShift Container Platform version 4.5 on the following IBM hardware:

- IBM Z: z13, z13s, all z14 models, all z15 models
- LinuxONE: all models

Hardware requirements

- 1 LPAR with 3 IFLs that supports SMT2
- 1 OSA or RoCE network adapter

Operating system requirements

- One instance of z/VM 7.1

On your z/VM instance, set up:

- 3 guest virtual machines for OpenShift Container Platform control plane machines
- 2 guest virtual machines for OpenShift Container Platform compute machines
- 1 guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

6.1.3.6. Preferred IBM Z system requirements

Hardware requirements

- 3 LPARs with 6 IFLs each that support SMT2
- 1 or 2 OSA or RoCE network adapters, or both
- Hipersockets, which are attached to a node either directly as a device or by bridging with one z/VM VSWITCH to be transparent to the z/VM guest. To directly connect Hipersockets to a node, you must set up a gateway to the external network via a RHEL 8 guest to bridge to the Hipersockets network.

Operating system requirements

- 2 or 3 instances of z/VM 7.1 for high availability

On your z/VM instances, set up:

- 3 guest virtual machines for OpenShift Container Platform control plane machines, one per z/VM instance
- At least 6 guest virtual machines for OpenShift Container Platform compute machines, distributed across the z/VM instances
- 1 guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV and High Performance FICON (zHPF) to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

6.1.3.7. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The

machine-approver cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

Additional resources

- See [Bridging a HiperSockets LAN with a z/VM Virtual Switch](#) in the IBM Knowledge Center.
- See [Scaling HyperPAV alias devices on Linux guests on z/VM](#) for performance optimization.

6.1.4. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Set up static IP addresses.
2. Set up an FTP server.
3. Provision the required load balancers.
4. Configure the ports for your machines.
5. Configure DNS.
6. Ensure network connectivity.

6.1.4.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 6.1. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .

Protocol	Port	Description
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 6.2. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 6.3. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



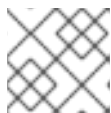
IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

**NOTE**

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 6.4. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 6.5. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic

Port	Back-end machines (pool members)	Internal	External	Description
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources

- [Configuring chrony time service](#)

6.1.4.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 6.6. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>..	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

Component	Record	Description
	api-int.<cluster_name>.<base_domain>	<p>Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.</p> </div> </div>
Routes	*.apps.<cluster_name>.<base_domain>	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 6.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
```

```

2019070700 ; serial
3H ; refresh (3 hours)
30M ; retry (30 minutes)
2W ; expiry (2 weeks)
1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 6.2. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H ; refresh (3 hours)
  30M ; retry (30 minutes)
  2W ; expiry (2 weeks)
  1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.

```

```

;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

6.1.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.



IMPORTANT

Do not skip this procedure in production environments where disaster recovery and debugging is required.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```

$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1

```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

6.1.6. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on your provisioning machine.

Prerequisites

- You must install the cluster from a machine that runs Linux, for example Red Hat Enterprise Linux 8.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

6.1.7. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

6.1.7.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

6.1.7.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

6.1.7.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH**.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```


After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

6.1.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

6.1.8.1. Sample install-config.yaml file for IBM Z

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
```

```

- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15

```

- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.



NOTE

Simultaneous multithreading (SMT) is enabled by default. If SMT is not enabled in your BIOS settings, the **hyperthreading** parameter has no effect.



IMPORTANT

If you disable **hyperthreading**, whether in the BIOS or in the **install-config.yaml**, ensure that your capacity planning accounts for the dramatically decreased machine performance.

- 4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.

- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control
- 8 The cluster name that you specified in your DNS records.
- 9 A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to access the pods from an external network, you must configure load balancers and routers to manage the traffic.
- 10 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for IBM Z infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 The pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

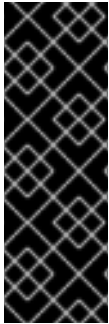


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

6.1.9. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** file.

- b. Locate the **mastersSchedulable** parameter and set its value to **False**.
 - c. Save and exit the file.
5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- 1 2 Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

6. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

6.1.10. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

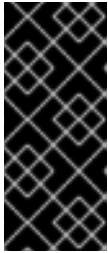
Before you install a cluster on IBM Z infrastructure that you provision, you must install RHCOS on z/VM guest virtual machines for the cluster to use. Complete the following steps to create the machines.

Prerequisites

- An FTP server running on your provisioning machine that is accessible to the machines you create.

Procedure

1. Log in to Linux on your provisioning machine.
2. Download the Red Hat Enterprise Linux CoreOS (RHCOS) installation files from the [RHCOS image mirror](#).



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

Download the following files:

- The initramfs: **rhcos-<version>-installer-initramfs.img**
 - The kernel: **rhcos-<version>-installer-kernel**
 - The operating system image for the disk on which you want to install RHCOS. This type can differ by virtual machine:
 - rhcos-<version>-s390x-dasd.s390x.raw.gz** for DASD
 - rhcos-<version>-s390x-metal.s390x.raw.gz** for FCP
3. Create parameter files. The following parameters are specific for a particular virtual machine:
 - For **coreos.inst.install_dev=**, specify **dasda** for a DASD installation, or **sda** for FCP. Note that FCP requires **zfcp.allow_lun_scan=0**.
 - For **rd.dasd=**, specifies the DASD where RHCOS is to be installed.
 - **rd.zfcp=<adapter>,<wwpn>,<lun>** specifies the FCP disk to install RHCOS on.
 - For **ip=**, specify the following seven entries:
 - i. The IP address for the machine.
 - ii. An empty string.
 - iii. The gateway.
 - iv. The netmask.
 - v. The machine host and domain name in the form **hostname.domainname**. Omit this value to let RHCOS decide set it.
 - vi. The network interface name. Omit this value to let RHCOS decide set it.
 - vii. If you use static IP addresses, an empty string.
 - For **coreos.inst.ignition_url=**, specify the Ignition file for the machine role. Use **bootstrap.ign**, **master.ign**, or **worker.ign**.
 - All other parameters can stay as they are.

Example parameter file, **bootstrap-0.parm**, for the bootstrap machine:

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasd coreos.inst.image_url=ftp://
cl1.provide.example.com:8080/assets/rhcos-43.80.20200430.0-s390x-dasd.390x.raw.gz
coreos.inst.ignition_url=ftp://cl1.provide.example.com:8080/ignition-bootstrap-0
ip=172.18.78.2::172.18.78.1:255.255.255.0::none nameserver=172.18.78.1
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 zfcplib.allow_lun_scan=0
cio_ignore=all,
!condev rd.dasd=0.0.3490
```

4. Transfer the initramfs, kernel, parameter files, and RHCOS images to z/VM, for example with FTP. For details about how to transfer the files with FTP and boot from the virtual reader, see [Installing under Z/VM](#).
5. Punch the files to the virtual reader of the z/VM guest virtual machine that is to become your bootstrap node.
See [PUNCH](#) in the IBM Knowledge Center.

TIP

You can use the CP PUNCH command or, if you use Linux, the **vmur** command to transfer files between two z/VM guest virtual machines.

6. Log in to CMS on the bootstrap machine.
7. IPL the bootstrap machine from the reader:

```
$ ipl c
```

See [IPL](#) in the IBM Knowledge Center.

8. Repeat this procedure for the other machines in the cluster.

6.1.11. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

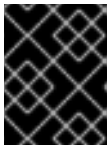
- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

6.1.12. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```


Example output

```
system:admin
```

6.1.13. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   63m   v1.18.3
master-1  Ready   master   63m   v1.18.3
master-2  Ready   master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                CONDITION
csr-mddf5  20m   system:node:master-01.example.com        Approved,Issued
csr-z5rln  16m   system:node:worker-21.example.com        Approved,Issued
```

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

**NOTE**

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready     master   73m   v1.20.0
master-1  Ready     master   73m   v1.20.0
master-2  Ready     master   74m   v1.20.0
worker-0  Ready     worker   11m   v1.20.0
worker-1  Ready     worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

6.1.14. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s

network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

6.1.14.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

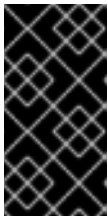
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

6.1.14.1.1. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

**NOTE**

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

**NOTE**

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

6.1.14.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```

**WARNING**

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

6.1.15. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m

machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE          NAME          READY STATUS
```

```

RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running  1      9m
openshift-apiserver           apiserver-67b9g                               1/1  Running  0
3m
openshift-apiserver           apiserver-ljcmx                               1/1  Running  0
1m
openshift-apiserver           apiserver-z25h4                               1/1  Running  0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8  1/1
Running  0      5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> 1
```

- 1 Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

6.1.16. Collecting debugging information

You can gather debugging information that might help you to troubleshoot and debug certain issues with an OpenShift Container Platform installation on IBM Z.

Prerequisites

- The **oc** CLI tool installed.

Procedure

1. Log in to the cluster:

```
$ oc login
```

2. On the node you want to gather hardware information about, start a debugging container:

```
$ oc debug node/<nodename>
```

3. Change to the **/host** file system and start **toolbox**:

```
$ chroot /host
$ toolbox
```

4. Collect the **dbginfo** data:

```
$ dbginfo.sh
```


- You can then retrieve the data, for example, using **scp**.

6.1.17. Additional resources

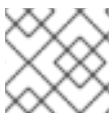
- See also [How to generate SOSREPORT within OpenShift4 nodes without SSH](#) .

6.1.18. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#) .

6.2. INSTALLING A CLUSTER ON IBM Z AND LINUXONE IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.5, you can install a cluster on IBM Z and LinuxONE infrastructure that you provision in a restricted network.



NOTE

While this document refers only to IBM Z, all information in it also applies to LinuxONE.



IMPORTANT

Additional considerations exist for non-bare metal platforms. Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you install an OpenShift Container Platform cluster.

Prerequisites

- [Create a mirror registry for installation in a restricted network](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.
- Before you begin the installation process, you must move or remove any existing installation files. This ensures that the required installation files are created and updated during the installation process.



IMPORTANT

Ensure that installation steps are done from a machine with access to the installation media.

- Provision [persistent storage](#) using NFS for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.



NOTE

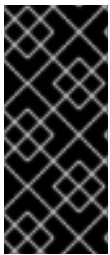
Be sure to also review this site list if you are configuring a proxy.

6.2.1. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

6.2.1.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

6.2.2. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

6.2.2.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To improve high availability of your cluster, distribute the control plane machines over different z/VM instances on at least two physical machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

6.2.2.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. The machines are configured with static IP addresses. No DHCP server is required. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server.

6.2.2.3. IBM Z network connectivity requirements

To install on IBM Z under z/VM, you require a single z/VM virtual NIC in layer 2 mode. You also need:

- A direct-attached OSA or RoCE network adapter
- A z/VM VSWITCH set up. For a preferred setup, use OSA link aggregation.

6.2.2.4. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS	2	8 GB	120 GB

1. 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: $(\text{threads per core} \times \text{cores}) \times \text{sockets} = \text{vCPUs}$.

6.2.2.5. Minimum IBM Z system requirements

You can install OpenShift Container Platform version 4.5 on the following IBM hardware:

- IBM Z: z13, z13s, all z14 models, all z15 models
- LinuxONE: all models

Hardware requirements

- 1 LPAR with 3 IFLs that supports SMT2
- 1 OSA or RoCE network adapter

Operating system requirements

- One instance of z/VM 7.1

On your z/VM instance, set up:

- 3 guest virtual machines for OpenShift Container Platform control plane machines
- 2 guest virtual machines for OpenShift Container Platform compute machines
- 1 guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

6.2.2.6. Preferred IBM Z system requirements

Hardware requirements

- 3 LPARs with 6 IFLs each that support SMT2
- 1 or 2 OSA or RoCE network adapters, or both
- Hipersockets, which are attached to a node either directly as a device or by bridging with one z/VM VSWITCH to be transparent to the z/VM guest. To directly connect Hipersockets to a node, you must set up a gateway to the external network via a RHEL 8 guest to bridge to the Hipersockets network.

Operating system requirements

- 2 or 3 instances of z/VM 7.1 for high availability

On your z/VM instances, set up:

- 3 guest virtual machines for OpenShift Container Platform control plane machines, one per z/VM instance
- At least 6 guest virtual machines for OpenShift Container Platform compute machines, distributed across the z/VM instances

- 1 guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV and High Performance FICON (zHPF) to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

6.2.2.7. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

Additional resources

- See [Bridging a HiperSockets LAN with a z/VM Virtual Switch](#) in the IBM Knowledge Center.
- See [Scaling HyperPAV alias devices on Linux guests on z/VM](#) for performance optimization.

6.2.3. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.

5. Ensure network connectivity.

6.2.3.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 6.7. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 6.8. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 6.9. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

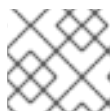
Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 6.10. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 6.11. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources

- [Configuring chrony time service](#)


6.2.3.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>.**

Table 6.12. Required DNS records

Component	Record	Description
-----------	--------	-------------

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.  IMPORTANT The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 6.3. Sample DNS zone database

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 6.4. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.

```

```

98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

6.2.4. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.



IMPORTANT

Do not skip this procedure in production environments where disaster recovery and debugging is required.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's **~/.ssh/authorized_keys** list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```

$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1

```

- 1 Specify the path and file name, such as **~/.ssh/id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

6.2.5. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

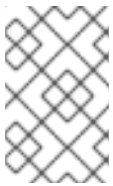
The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

6.2.5.1. Sample **install-config.yaml** file for IBM Z

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
```


- 9 A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to
- 10 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for IBM Z infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 16 Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.
- 17 Provide the **imageContentSources** section from the output of the command to mirror the repository.

6.2.5.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

6.2.6. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:

- Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` file.
- Locate the `mastersSchedulable` parameter and set its value to **False**.
- Save and exit the file.

- Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yaml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}
```

- ❶ ❷ Remove this section completely.

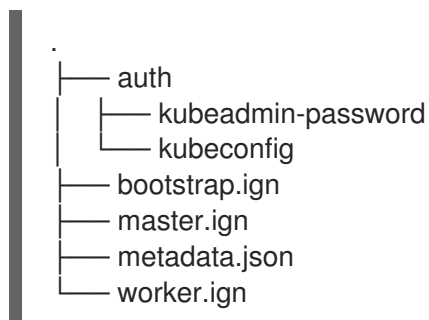
If you do so, you must add ingress DNS records manually in a later step.

- Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> ❶
```

- ❶ For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:



6.2.7. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on IBM Z infrastructure that you provision, you must install RHCOS on z/VM guest virtual machines for the cluster to use. Complete the following steps to create the machines.

Prerequisites

- An FTP server running on your provisioning machine that is accessible to the machines you create.

Procedure

1. Log in to Linux on your provisioning machine.
2. Download the Red Hat Enterprise Linux CoreOS (RHCOS) installation files from the [RHCOS image mirror](#).



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

Download the following files:

- The initramfs: **rhcos-<version>-installer-initramfs.img**
 - The kernel: **rhcos-<version>-installer-kernel**
 - The operating system image for the disk on which you want to install RHCOS. This type can differ by virtual machine:
rhcos-<version>-s390x-dasd.s390x.raw.gz for DASD
rhcos-<version>-s390x-metal.s390x.raw.gz for FCP
3. Create parameter files. The following parameters are specific for a particular virtual machine:
 - For **coreos.inst.install_dev=**, specify **dasda** for a DASD installation, or **sda** for FCP. Note that FCP requires **zfcp.allow_lun_scan=0**.
 - For **rd.dasd=**, specifies the DASD where RHCOS is to be installed.
 - **rd.zfcp=<adapter>,<wwpn>,<lun>** specifies the FCP disk to install RHCOS on.

- For **ip=**, specify the following seven entries:
 - i. The IP address for the machine.
 - ii. An empty string.
 - iii. The gateway.
 - iv. The netmask.
 - v. The machine host and domain name in the form **hostname.domainname**. Omit this value to let RHCOS decide set it.
 - vi. The network interface name. Omit this value to let RHCOS decide set it.
 - vii. If you use static IP addresses, an empty string.
- For **coreos.inst.ignition_url=**, specify the Ignition file for the machine role. Use **bootstrap.ign**, **master.ign**, or **worker.ign**.
- All other parameters can stay as they are.
Example parameter file, **bootstrap-0.parm**, for the bootstrap machine:

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda coreos.inst.image_url=ftp://
cl1.provide.example.com:8080/assets/rhcos-43.80.20200430.0-s390x-dasd.390x.raw.gz
coreos.inst.ignition_url=ftp://cl1.provide.example.com:8080/ignition-bootstrap-0
ip=172.18.78.2::172.18.78.1:255.255.255.0::none nameserver=172.18.78.1
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 zfcpl.allow_lun_scan=0
cio_ignore=all,
lcondev rd.dasd=0.0.3490
```

4. Transfer the initramfs, kernel, parameter files, and RHCOS images to z/VM, for example with FTP. For details about how to transfer the files with FTP and boot from the virtual reader, see [Installing under Z/VM](#).
5. Punch the files to the virtual reader of the z/VM guest virtual machine that is to become your bootstrap node.
See [PUNCH](#) in the IBM Knowledge Center.

TIP

You can use the CP PUNCH command or, if you use Linux, the **vmur** command to transfer files between two z/VM guest virtual machines.

6. Log in to CMS on the bootstrap machine.
7. IPL the bootstrap machine from the reader:

```
$ ipl c
```

See [IPL](#) in the IBM Knowledge Center.

8. Repeat this procedure for the other machines in the cluster.

6.2.8. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

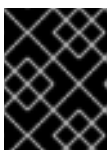
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

6.2.9. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

6.2.10. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-8b2br	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
csr-8vnps	15m	system:serviceaccount:openshift-machine-config-operator:node-bootstrapper	Pending
...			

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

NAME	AGE	REQUESTOR	CONDITION
csr-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal	Pending
csr-c57lv	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal	Pending
...			

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}\n{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

6.2.11. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

- Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

6.2.11.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

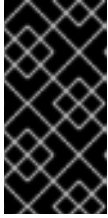
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

6.2.11.1.1. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

**IMPORTANT**

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

**NOTE**

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

**NOTE**

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

6.2.11.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:



```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

6.2.12. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME                                VERSION AVAILABLE PROGRESSING DEGRADED
```

SINCE					
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE          NAME                                     READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running    1    9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8    1/1
Running    0    5m
...
```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> ❶
```

- ❶ Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

3. Register your cluster on the [Cluster registration](#) page.

6.2.13. Collecting debugging information

You can gather debugging information that might help you to troubleshoot and debug certain issues with an OpenShift Container Platform installation on IBM Z.

Prerequisites

- The **oc** CLI tool installed.

Procedure

1. Log in to the cluster:

```
$ oc login
```

2. On the node you want to gather hardware information about, start a debugging container:

```
$ oc debug node/<nodename>
```

3. Change to the **/host** file system and start **toolbox**:

```
$ chroot /host  
$ toolbox
```

4. Collect the **dbginfo** data:

```
$ dbginfo.sh
```

5. You can then retrieve the data, for example, using **scp**.

Additional resources

- See also [How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH](#).

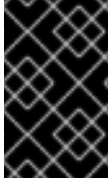
Next steps

- [Customize your cluster](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).

CHAPTER 7. INSTALLING ON IBM POWER

7.1. INSTALLING A CLUSTER ON IBM POWER

In OpenShift Container Platform version 4.5, you can install a cluster on IBM Power infrastructure that you provision.

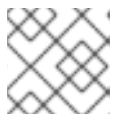


IMPORTANT

Additional considerations exist for non-bare metal platforms. Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you install an OpenShift Container Platform cluster.

Prerequisites

- Before you begin the installation process, you must move or remove any existing installation files. This ensures that the required installation files are created and updated during the installation process.
- Provision [persistent storage using NFS](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

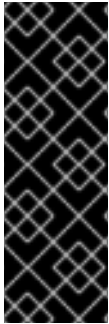
7.1.1. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

7.1.2. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

7.1.2.1. Required machines

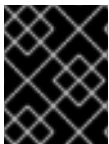
The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

7.1.2.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

7.1.2.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	2	16 GB	120 GB
Control plane	RHCOS	2	16 GB	120 GB
Compute	RHCOS	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

7.1.2.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

7.1.3. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

7.1.3.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 7.1. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 7.2. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 7.3. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



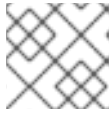
IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

**NOTE**

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 7.4. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 7.5. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources

- [Configuring chrony time service](#)


7.1.3.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 7.6. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

Component	Record	Description
	api-int.<cluster_name>.<base_domain>.	<p>Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.</p> </div> </div>
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 7.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H ; refresh (3 hours)
  30M ; retry (30 minutes)
```

```

2W ; expiry (2 weeks)
1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 7.2. Sample DNS zone database for reverse records

```

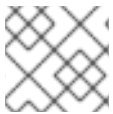
$TTL 1W
@ IN SOA ns1.example.com. root (
2019070700 ; serial
3H ; refresh (3 hours)
30M ; retry (30 minutes)
2W ; expiry (2 weeks)
1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;

```

```
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

7.1.4. Generating an SSH private key and adding it to the agent

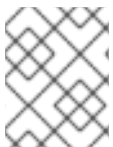
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

-

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

7.1.5. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.

**IMPORTANT**

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

**IMPORTANT**

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

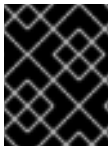
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

7.1.6. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

7.1.6.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

7.1.6.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

7.1.6.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

7.1.7. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

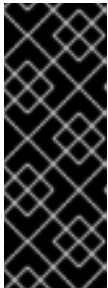
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

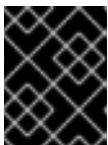
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

7.1.7.1. Sample install-config.yaml file for IBM Power

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
name: worker
```

```

replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15

```

- 1** The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2** **5** The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3** **6** Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.



NOTE

Simultaneous multithreading (SMT) is enabled by default. If SMT is not enabled in your BIOS settings, the **hyperthreading** parameter has no effect.



IMPORTANT

If you disable **hyperthreading**, whether in the BIOS or in the **install-config.yaml**, ensure that your capacity planning accounts for the dramatically decreased machine performance.

- 4** You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.
- 7** The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.

- 8 The cluster name that you specified in your DNS records.
- 9 A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to access the pods from an external network, you must configure load balancers and routers to manage the traffic.
- 10 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for IBM Power infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 The pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

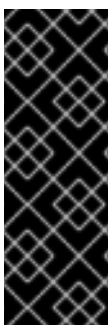


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

7.1.8. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Modify the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:

- a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yaml** file.
- b. Locate the **mastersSchedulable** parameter and set its value to **False**.
- c. Save and exit the file.

5. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yaml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
```

```

metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: ❶
    id: mycluster-100419-private-zone
  publicZone: ❷
    id: example.openshift.com
status: {}

```

❶ ❷ Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

6. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> ❶
```

❶ For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```

.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign

```

7.1.9. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on IBM Power infrastructure that you provision, you must create RHCOS machines for it to use. Follow either the steps to use an ISO image or network PXE booting to create the machines.

7.1.9.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

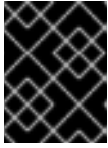
Before you install a cluster on IBM Power infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.

Procedure

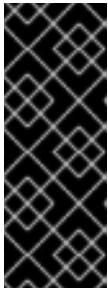
1. Upload the control plane, compute, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the [RHCOS image mirror](#) page.

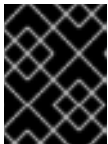


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use ISO images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

You must download the ISO file and the RAW disk file. Those file names resemble the following examples:

- ISO: **rhcos-<version>-installer.<architecture>.iso**
 - Compressed metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**
3. Upload either the RAW RHCOS image file to your HTTP server and note its URL.



IMPORTANT

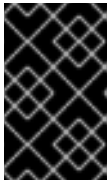
If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

4. Use the ISO to start the RHCOS installation. Use one of the following installation options:
 - Burn the ISO image to a disk and boot it directly.
 - Use ISO redirection via a LOM interface.
5. After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
6. Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

- 1** Specify the block device of the system to install to.

2. Specify the URL of the RAW image that you uploaded to your server.
 3. Specify the URL of the Ignition config file for this machine type.
 4. Set **ip=dhcp** or set an individual static IP address (**ip=**) and DNS server (**nameserver=**) on each node. See *Configure advanced networking* for details.
 5. If you use multiple network interfaces or DNS servers, see *Configure advanced networking* for details on how to configure them.
 6. Optionally, you can bond multiple network interfaces to a single interface using the **bond=** option, as described in *Configure advanced networking*.
7. Press Enter to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
 8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

7.1.9.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE booting

Before you install a cluster on IBM Power infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server and TFTP server that you can access from your computer.

Procedure

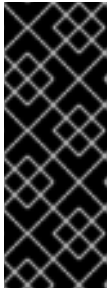
1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the compressed metal RAW image, **kernel** and **initramfs** files from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.

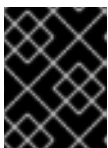


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use RAW images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- Compressed metal RAW image: **rhcos-`<version>`-`<architecture>`-metal.`<architecture>`.raw.gz**
 - **kernel:** **rhcos-`<version>`-`<architecture>`-installer-kernel-`<architecture>`**
 - **initramfs:** **rhcos-`<version>`-`<architecture>`-installer-initramfs.`<architecture>`.img**
3. Upload the RAW image to your HTTP server.
 4. Upload the additional files that are required for your booting method:
 - For traditional PXE, upload the **kernel** and **initramfs** files to your TFTP server.
 - For iPXE, upload the **kernel** and **initramfs** files to your HTTP server.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
6. Configure PXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```

DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
  KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> 1
  APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2
3

```

- 1** Specify the location of the **kernel** file available on your TFTP server.

2

If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

- 3 Specify locations of the RHCOS files that you uploaded to your HTTP or TFTP server. The **initrd** parameter value is the location of the **initramfs** file on your TFTP server. The **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image on your HTTP server, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file on your HTTP server.



NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

7. If you use UEFI, perform the following actions:

- a. Provide the EFI binaries and **grub.cfg** file that are required for booting the system. You need the **shim.efi** binary and the **grubx64.efi** binary.
 - Extract the necessary EFI binaries by mounting the RHCOS ISO on your host and then mounting the **images/efiboot.img** file to your host. From the **efiboot.img** mount point, you then copy the **EFI/redhat/shimx64.efi** and **EFI/redhat/grubx64.efi** files to your TFTP server.

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

- b. Copy the **EFI/redhat/grub.cfg** file that is included in the RHCOS ISO to your TFTP server.
- c. Edit the **grub.cfg** file to include the following arguments:

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
  linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
  rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
  coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
  <architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
  initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

- 1 The first argument to the **linux** line item is the location of the **kernel** file that you uploaded to your TFTP server. For the **coreos.inst.image_url** parameter value, specify the location of the compressed metal RAW image that you uploaded to your HTTP server. For the **coreos.inst.ignition_url** parameter, specify the location of the bootstrap Ignition config file that you uploaded to your HTTP server.

- 2 Specify the location of the **iniramfs** file that you uploaded to your TFTP server.

8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

7.1.10. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

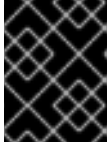
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

7.1.11. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

7.1.12. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```

NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3

```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}\n' | xargs --no-run-if-empty oc adm certificate approve
```

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

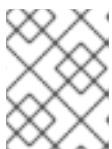
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   73m   v1.20.0
master-1  Ready   master   73m   v1.20.0
master-2  Ready   master   74m   v1.20.0
worker-0  Ready   worker   11m   v1.20.0
worker-1  Ready   worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

7.1.13. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

7.1.13.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

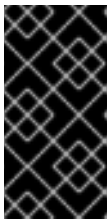
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

7.1.13.1.1. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.



NOTE

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

7.1.13.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

7.1.14. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.

- a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE          NAME                                     READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running    1    9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0    5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> 1
```

- 1** Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

7.2. INSTALLING A CLUSTER ON IBM POWER IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.5, you can install a cluster on IBM Power infrastructure that you provision in a restricted network.

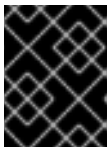


IMPORTANT

Additional considerations exist for non-bare metal platforms. Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you install an OpenShift Container Platform cluster.

Prerequisites

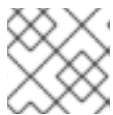
- [Create a mirror registry for installation in a restricted network](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform. Use the **docker.io/ibmcom/registry-ppc64le:2.6.2.5** image.
- Before you begin the installation process, you must move or remove any existing installation files. This ensures that the required installation files are created and updated during the installation process.



IMPORTANT

Ensure that installation steps are done from a machine with access to the installation media.

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

7.2.1. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods

that meet your restrictions.



IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

7.2.1.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

7.2.2. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

7.2.2.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines.



NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.



IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

7.2.2.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines

require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

7.2.2.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	2	16 GB	120 GB
Control plane	RHCOS	2	16 GB	120 GB
Compute	RHCOS	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: $(\text{threads per core} \times \text{cores}) \times \text{sockets} = \text{vCPUs}$.

7.2.2.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

7.2.3. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.

5. Ensure network connectivity.

7.2.3.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 7.7. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 7.8. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 7.9. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 7.10. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 7.11. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Additional resources

- [Configuring chrony time service](#)


7.2.3.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 7.12. Required DNS records

Component	Record	Description
-----------	--------	-------------

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.  IMPORTANT The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 7.3. Sample DNS zone database

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 7.4. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.

```

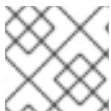
```

98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

7.2.4. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

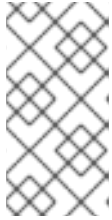
```

$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1

```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

7.2.5. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

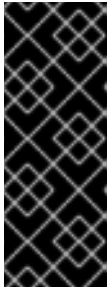
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

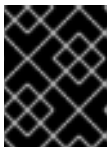
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

7.2.5.1. Sample install-config.yaml file for IBM Power

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
```


- 9 A block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to
- 10 The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IPs addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- 11 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- 12 You must set the platform to **none**. You cannot provide additional platform configuration variables for IBM Power infrastructure.
- 13 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 14 For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.
- 15 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 16 Provide the contents of the certificate file that you used for your mirror registry.
- 17 Provide the **imageContentSources** section from the output of the command to mirror the repository.

7.2.5.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

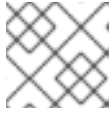
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

7.2.6. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
 - Locate the `mastersSchedulable` parameter and set its value to **False**.
 - Save and exit the file.
- Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- Remove this section completely.

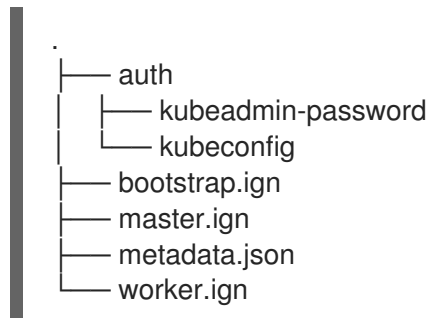
If you do so, you must add ingress DNS records manually in a later step.

- Obtain the Ignition config files:

```
./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:



7.2.7. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on IBM Power infrastructure that you provision, you must create RHCOS machines for it to use. Follow either the steps to use an ISO image or network PXE booting to create the machines.

7.2.7.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

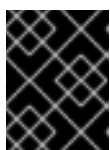
Before you install a cluster on IBM Power infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.

Procedure

1. Upload the control plane, compute, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the [RHCOS image mirror](#) page.

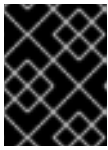


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use ISO images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

You must download the ISO file and the RAW disk file. Those file names resemble the following examples:

- ISO: **rhcos-<version>-installer.<architecture>.iso**
 - Compressed metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**
3. Upload either the RAW RHCOS image file to your HTTP server and note its URL.



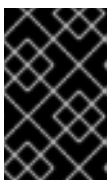
IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

4. Use the ISO to start the RHCOS installation. Use one of the following installation options:
 - Burn the ISO image to a disk and boot it directly.
 - Use ISO redirection via a LOM interface.
5. After the instance boots, press the **TAB** or **E** key to edit the kernel command line.
6. Add the parameters to the kernel command line:

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

- 1 Specify the block device of the system to install to.
 - 2 Specify the URL of the RAW image that you uploaded to your server.
 - 3 Specify the URL of the Ignition config file for this machine type.
 - 4 Set **ip=dhcp** or set an individual static IP address (**ip=**) and DNS server (**nameserver=**) on each node. See *Configure advanced networking* for details.
 - 5 If you use multiple network interfaces or DNS servers, see *Configure advanced networking* for details on how to configure them.
 - 6 Optionally, you can bond multiple network interfaces to a single interface using the **bond=** option, as described in *Configure advanced networking*.
7. Press Enter to complete the installation. After RHCOS installs, the system reboots. After the system reboots, it applies the Ignition config file that you specified.
 8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

7.2.7.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE booting

Before you install a cluster on IBM Power infrastructure that you provision, you must create RHCOS machines for it to use. You can use PXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server and TFTP server that you can access from your computer.

Procedure

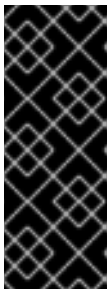
1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. Obtain the compressed metal RAW image, **kernel** and **initramfs** files from the [Product Downloads](#) page on the Red Hat customer portal or the [RHCOS image mirror](#) page.

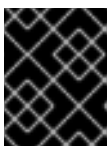


IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Only use RAW images for this procedure. RHCOS qcow2 images are not supported for bare metal installs.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- Compressed metal RAW image: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
 - **kernel: rhcos-<version>-<architecture>-installer-kernel-<architecture>**
 - **initramfs: rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**
3. Upload the RAW image to your HTTP server.
 4. Upload the additional files that are required for your booting method:
 - For traditional PXE, upload the **kernel** and **initramfs** files to your TFTP server.
 - For iPXE, upload the **kernel** and **initramfs** files to your HTTP server.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
6. Configure PXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```

DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
  KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> 1
  APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 2
3

```

- 1** Specify the location of the **kernel** file available on your TFTP server.
- 2** If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.
- 3** Specify locations of the RHCOS files that you uploaded to your HTTP or TFTP server. The **initrd** parameter value is the location of the **initramfs** file on your TFTP server. The **coreos.inst.image_url** parameter value is the location of the compressed metal RAW image on your HTTP server, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file on your HTTP server.



NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see [How does one set up a serial terminal and/or console in Red Hat Enterprise Linux?](#)

7. If you use UEFI, perform the following actions:
 - a. Provide the EFI binaries and **grub.cfg** file that are required for booting the system. You need the **shim.efi** binary and the **grubx64.efi** binary.
 - Extract the necessary EFI binaries by mounting the RHCOS ISO on your host and then mounting the **images/efiboot.img** file to your host. From the **efiboot.img** mount point, you then copy the **EFI/redhat/shimx64.efi** and **EFI/redhat/grubx64.efi** files to your TFTP server.

```

# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot

```



```
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

- b. Copy the **EFI/redhat/grub.cfg** file that is included in the RHCOS ISO to your TFTP server.
- c. Edit the **grub.cfg** file to include the following arguments:

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
  linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
  rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
  coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
  <architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
  initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

- 1** The first argument to the **linux** line item is the location of the **kernel** file that you uploaded to your TFTP server. For the **coreos.inst.image_url** parameter value, specify the location of the compressed metal RAW image that you uploaded to your HTTP server. For the **coreos.inst.ignition_url** parameter, specify the location of the bootstrap Ignition config file that you uploaded to your HTTP server.
- 2** Specify the location of the **initramfs** file that you uploaded to your TFTP server.

8. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, which is the default, also create at least two compute machines before you install the cluster.

7.2.8. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

7.2.9. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

7.2.10. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   63m   v1.18.3
master-1  Ready   master   63m   v1.18.3
master-2  Ready   master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE    VERSION
master-0  Ready    master   73m    v1.20.0
master-1  Ready    master   73m    v1.20.0
master-2  Ready    master   74m    v1.20.0
worker-0  Ready    worker   11m    v1.20.0
worker-1  Ready    worker   11m    v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

7.2.11. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s

network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

7.2.11.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

7.2.11.1.1. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

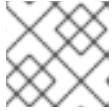
1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

**NOTE**

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

**NOTE**

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
pvc:
claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

7.2.11.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```

**WARNING**

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

7.2.12. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m

machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

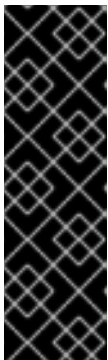
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE          NAME          READY STATUS
```

```
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running  1      9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8  1/1
Running  0      5m
...
```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> 1
```

- 1 Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

Next steps

- [Customize your cluster.](#)
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores.](#)

CHAPTER 8. INSTALLING ON OPENSTACK

8.1. INSTALLING A CLUSTER ON OPENSTACK WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a customized cluster on Red Hat OpenStack Platform (RHOSP). To customize the installation, modify parameters in the **install-config.yaml** before you install the cluster.

8.1.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.5 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).
- Verify that your network configuration does not rely on a provider network. Provider networks are not supported.
- Have a storage service installed in RHOSP, like block storage (Cinder) or object storage (Swift). Object storage is the recommended storage technology for OpenShift Container Platform registry cluster deployment. For more information, see [Optimizing storage](#).
- Have metadata service enabled in RHOSP

8.1.2. Resource guidelines for installing OpenShift Container Platform on RHOSP

To support an OpenShift Container Platform installation, your Red Hat OpenStack Platform (RHOSP) quota must meet the following requirements:

Table 8.1. Recommended resources for a default OpenShift Container Platform cluster on RHOSP

Resource	Value
Floating IP addresses	3
Ports	15
Routers	1
Subnets	1
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7

Resource	Value
Security groups	3
Security group rules	60

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If RHOSP object storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



NOTE

By default, your security group and security group rule quotas might be low. If you encounter problems, run **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** as an administrator to increase them.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

8.1.2.1. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

8.1.2.2. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota

- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

8.1.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

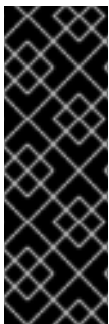


IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

8.1.4. Enabling Swift on RHOSP

Swift is operated by a user account with the **swiftoperator** role. Add the role to an account before you run the installation program.



IMPORTANT

If [the Red Hat OpenStack Platform \(RHOSP\) object storage service](#), commonly known as Swift, is available, OpenShift Container Platform uses it as the image registry storage. If it is unavailable, the installation program relies on the RHOSP block storage service, commonly known as Cinder.

If Swift is present and you want to use it, you must enable access to it. If it is not present, or if you do not want to use it, skip this section.

Prerequisites

- You have a RHOSP administrator account on the target environment.
- The Swift service is installed.

- On [Ceph RGW](#), the **account in url** option is enabled.

Procedure

To enable Swift on RHOSP:

1. As an administrator in the RHOSP CLI, add the **swiftoperator** role to the account that will access Swift:

```
$ openstack role add --user <user> --project <project> swiftoperator
```

Your RHOSP deployment can now use Swift for the image registry.

8.1.5. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the external router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries](#)

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

Example output

```
+-----+-----+-----+
| ID                | Name      | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External   |
+-----+-----+-----+
```

A network with an external router type appears in the network list. If at least one does not, see [Creating a default floating IP network](#) and [Creating a default provider network](#).

IMPORTANT

If the external network's CIDR range overlaps one of the default network ranges, you must change the matching network ranges in the **install-config.yaml** file before you start the installation process.

The default network ranges are:

Network	Range
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

**WARNING**

If the installation program finds multiple networks with the same name, it sets one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

**NOTE**

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

8.1.6. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:
 - If your RHOSP distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.

**IMPORTANT**

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your RHOSP distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.



```

clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'

```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:

- a. Copy the certificate authority file to your machine.
- b. Add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

- c. Update the trust bundle:

```
$ sudo update-ca-trust extract
```

- d. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```

clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"

```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**

- d. A Unix-specific site configuration directory, for example `/etc/openstack/clouds.yaml`
The installation program searches for `clouds.yaml` in that order.

8.1.7. Obtaining the installation program

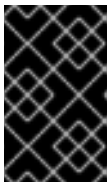
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a `.txt` file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

8.1.8. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- v. Select the region to deploy the cluster to.
- vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- vii. Select **openstack** as the platform to target.
- viii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
- ix. Specify the floating IP address to use for external access to the OpenShift API.

- x. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - xi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - xii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - xiii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

8.1.8.1. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
```

```

noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...

```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.



NOTE

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

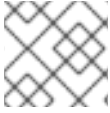


NOTE

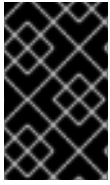
Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

8.1.9. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

8.1.9.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 8.2. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev . The string must be 14 characters or fewer long.


Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

8.1.9.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 8.3. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	<p>Object</p>  <p>NOTE</p> <p>You cannot modify parameters specified by the networking object after installation.</p>

Parameter	Description	Values
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.



8.1.9.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 8.4. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

8.1.9.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters

Additional RHOSP configuration parameters are described in the following table:

Table 8.5. Additional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .

8.1.9.5. Optional RHOSP configuration parameters

Optional RHOSP configuration parameters are described in the following table:

Table 8.6. Optional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.additionalNetworkIDs	Additional networks that are associated with compute machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
compute.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with compute machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
controlPlane.platform.openstack.additionalNetworkIDs	Additional networks that are associated with control plane machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
controlPlane.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with control plane machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
platform.openstack.clusterOSImage	<p>The location from which the installer downloads the RHCOS image.</p> <p>You must set this parameter to perform an installation in a restricted network.</p>	<p>An HTTP or HTTPS URL, optionally with an SHA-256 checksum.</p> <p>For example, http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d.</p> <p>The value can also be the name of an existing Glance image, for example my-rhcos.</p>
platform.openstack.defaultMachinePlatform	The default machine pool platform configuration.	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>
platform.openstack.externalDNS	IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings. For example, ["8.8.8.8", "192.168.1.12"] .

Parameter	Description	Values
platform.openstack.machinesSubnet	<p>The UUID of a RHOSP subnet that the cluster's nodes use. Nodes and virtual IP (VIP) ports are created on this subnet.</p> <p>The first item in networking.machineNetwork must match the value of machinesSubnet.</p> <p>If you deploy to a custom subnet, you cannot specify an external DNS server to the OpenShift Container Platform installer. Instead, add DNS to the subnet in RHOSP.</p>	A UUID as a string, for example fa806b2f-ac49-4bce-b9db-124bc64209bf .

8.1.9.6. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 8.7. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

8.1.9.7. Custom subnets in RHOSP deployments

Optionally, you can deploy a cluster on a Red Hat OpenStack Platform (RHOSP) subnet of your choice. The subnet's GUID is passed as the value of **platform.openstack.machinesSubnet** in the **install-config.yaml** file.

This subnet is used as the cluster's primary subnet; nodes and ports are created on it.

Before you run the OpenShift Container Platform installer with a custom subnet, verify that:

- The target network and subnet are available.
- DHCP is enabled on the target subnet.
- You can provide installer credentials that have permission to create ports on the target network.
- If your network configuration requires a router, it is created in RHOSP. Some configurations rely on routers for floating IP address translation.
- Your network configuration does not rely on a provider network. Provider networks are not supported.

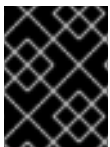


NOTE

By default, the API VIP takes x.x.x.5 and the Ingress VIP takes x.x.x.7 from your network's CIDR block. To override these default values, set values for **platform.openstack.apiVIP** and **platform.openstack.ingressVIP** that are outside of the DHCP allocation pool.

8.1.9.8. Sample customized `install-config.yaml` file for RHOSP

This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

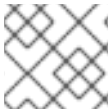
This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
```

```
- 172.30.0.0/16
networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

8.1.10. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output


```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

8.1.11. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible with or without floating IP addresses.

8.1.11.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.



IMPORTANT

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

- Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

- To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```



NOTE

If you do not control the DNS server you can add the record to your `/etc/hosts` file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

8.1.11.2. Enabling access without floating IP addresses

If you cannot use floating IP addresses, the OpenShift Container Platform installation might still finish. However, the installation program fails after it times out waiting for API access.

After the installation program times out, the cluster might still initialize. After the bootstrapping processing begins, it must complete. You must edit the cluster's networking configuration after it is deployed.

8.1.12. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

- Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

- The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
- The `~/.gcp/osServiceAccount.json` file
- The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For `<installation_directory>`, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

8.1.13. Verifying cluster status

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:

```
$ oc get clusterversion
```

4. View your Operators' status:

```
$ oc get clusteroperator
```

5. View all running pods in the cluster:

```
$ oc get pods -A
```

8.1.14. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

■

```
$ oc whoami
```

Example output

```
system:admin
```

8.1.15. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.1.16. Next steps

- [Customize your cluster.](#)

- If necessary, you can [opt out of remote health reporting](#) .
- If you need to enable external access to node ports, [configure ingress cluster traffic by using a node port](#).

8.2. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR

In OpenShift Container Platform version 4.5, you can install a customized cluster on Red Hat OpenStack Platform (RHOSP) that uses Kuryr SDN. To customize the installation, modify parameters in the **install-config.yaml** before you install the cluster.

8.2.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.5 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#) .
- Verify that your network configuration does not rely on a provider network. Provider networks are not supported.
- Have a storage service installed in RHOSP, like block storage (Cinder) or object storage (Swift). Object storage is the recommended storage technology for OpenShift Container Platform registry cluster deployment. For more information, see [Optimizing storage](#) .

8.2.2. About Kuryr SDN

[Kuryr](#) is a container network interface (CNI) plug-in solution that uses the [Neutron](#) and [Octavia](#) Red Hat OpenStack Platform (RHOSP) services to provide networking for pods and Services.

Kuryr and OpenShift Container Platform integration is primarily designed for OpenShift Container Platform clusters running on RHOSP VMs. Kuryr improves the network performance by plugging OpenShift Container Platform pods into RHOSP SDN. In addition, it provides interconnectivity between pods and RHOSP virtual instances.

Kuryr components are installed as pods in OpenShift Container Platform using the **openshift-kuryr** namespace:

- **kuryr-controller** - a single service instance installed on a **master** node. This is modeled in OpenShift Container Platform as a **Deployment** object.
- **kuryr-cni** - a container installing and configuring Kuryr as a CNI driver on each OpenShift Container Platform node. This is modeled in OpenShift Container Platform as a **DaemonSet** object.

The Kuryr controller watches the OpenShift Container Platform API server for pod, service, and namespace create, update, and delete events. It maps the OpenShift Container Platform API calls to corresponding objects in Neutron and Octavia. This means that every network solution that implements the Neutron trunk port functionality can be used to back OpenShift Container Platform via Kuryr. This includes open source solutions such as Open vSwitch (OVS) and Open Virtual Network (OVN) as well as Neutron-compatible commercial SDNs.

Kuryr is recommended for OpenShift Container Platform deployments on encapsulated RHOSP tenant networks to avoid double encapsulation, such as running an encapsulated OpenShift Container Platform SDN over an RHOSP network.

If you use provider networks or tenant VLANs, you do not need to use Kuryr to avoid double encapsulation. The performance benefit is negligible. Depending on your configuration, though, using Kuryr to avoid having two overlays might still be beneficial.

Kuryr is not recommended in deployments where all of the following criteria are true:

- The RHOSP version is less than 16.
- The deployment uses UDP services, or a large number of TCP services on few hypervisors.

or

- The **ovn-octavia** Octavia driver is disabled.
- The deployment uses a large number of TCP services on few hypervisors.

8.2.3. Resource guidelines for installing OpenShift Container Platform on RHOSP with Kuryr

When using Kuryr SDN, the pods, services, namespaces, and network policies are using resources from the RHOSP quota; this increases the minimum requirements. Kuryr also has some additional requirements on top of what a default install requires.

Use the following quota to satisfy a default cluster's minimum requirements:

Table 8.8. Recommended resources for a default OpenShift Container Platform cluster on RHOSP with Kuryr

Resource	Value
Floating IP addresses	3 - plus the expected number of Services of LoadBalancer type
Ports	1500 - 1 needed per Pod
Routers	1
Subnets	250 - 1 needed per Namespace/Project
Networks	250 - 1 needed per Namespace/Project
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	250 - 1 needed per Service and per NetworkPolicy

Resource	Value
Security group rules	1000
Load balancers	100 - 1 needed per Service
Load balancer listeners	500 - 1 needed per Service-exposed port
Load balancer pools	500 - 1 needed per Service-exposed port

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If RHOSP object storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



IMPORTANT

If you are using Red Hat OpenStack Platform (RHOSP) version 16 with the Amphora driver rather than the OVN Octavia driver, security groups are associated with service accounts instead of user projects.

Take the following notes into consideration when setting resources:

- The number of ports that are required is larger than the number of pods. Kuryr uses ports pools to have pre-created ports ready to be used by pods and speed up the pods' booting time.
- Each network policy is mapped into an RHOSP security group, and depending on the **NetworkPolicy** spec, one or more rules are added to the security group.
- Each service is mapped to an RHOSP load balancer. Consider this requirement when estimating the number of security groups required for the quota.
If you are using RHOSP version 15 or earlier, or the **ovn-octavia driver**, each load balancer has a security group with the user project.
- The quota does not account for load balancer resources (such as VM resources), but you must consider these resources when you decide the RHOSP deployment's size. The default installation will have more than 50 load balancers; the clusters must be able to accommodate them.
If you are using RHOSP version 16 with the OVN Octavia driver enabled, only one load balancer VM is generated; services are load balanced through OVN flows.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

To enable Kuryr SDN, your environment must meet the following requirements:

- Run RHOSP 13+.

- Have Overcloud with Octavia.
- Use Neutron Trunk ports extension.
- Use **openvswitch** firewall driver if ML2/OVS Neutron driver is used instead of **ovs-hybrid**.

8.2.3.1. Increasing quota

When using Kuryr SDN, you must increase quotas to satisfy the Red Hat OpenStack Platform (RHOSP) resources used by pods, services, namespaces, and network policies.

Procedure

- Increase the quotas for a project by running the following command:

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets 250 --networks 250 <project>
```

8.2.3.2. Configuring Neutron

Kuryr CNI leverages the Neutron Trunks extension to plug containers into the Red Hat OpenStack Platform (RHOSP) SDN, so you must use the **trunks** extension for Kuryr to properly work.

In addition, if you leverage the default ML2/OVS Neutron driver, the firewall must be set to **openvswitch** instead of **ovs_hybrid** so that security groups are enforced on trunk subports and Kuryr can properly handle network policies.

8.2.3.3. Configuring Octavia

Kuryr SDN uses Red Hat OpenStack Platform (RHOSP)'s Octavia LBaaS to implement OpenShift Container Platform services. Thus, you must install and configure Octavia components in RHOSP to use Kuryr SDN.

To enable Octavia, you must include the Octavia service during the installation of the RHOSP Overcloud, or upgrade the Octavia service if the Overcloud already exists. The following steps for enabling Octavia apply to both a clean install of the Overcloud or an Overcloud update.



NOTE

The following steps only capture the key pieces required during the [deployment of RHOSP](#) when dealing with Octavia. It is also important to note that [registry methods](#) vary.

This example uses the local registry method.

Procedure

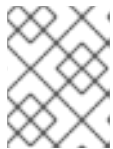
1. If you are using the local registry, create a template to upload the images to the registry. For example:

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack-
```

```
--tag-from-label {version}-{release} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. Verify that the **local_registry_images.yaml** file contains the Octavia images. For example:

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```



NOTE

The Octavia container versions vary depending upon the specific RHOSP release installed.

3. Pull the container images from **registry.redhat.io** to the Undercloud node:

```
(undercloud) $ sudo openstack overcloud container image upload \
--config-file /home/stack/local_registry_images.yaml \
--verbose
```

This may take some time depending on the speed of your network and Undercloud disk.

4. Since an Octavia load balancer is used to access the OpenShift Container Platform API, you must increase their listeners' default timeouts for the connections. The default timeout is 50 seconds. Increase the timeout to 20 minutes by passing the following file to the Overcloud deploy command:

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```



NOTE

This is not needed for RHOSP 13.0.13+.

5. Install or update your Overcloud environment with Octavia:

```
$ openstack overcloud deploy --templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
-e octavia_timeouts.yaml
```

**NOTE**

This command only includes the files associated with Octavia; it varies based on your specific installation of RHOSP. See the RHOSP documentation for further information. For more information on customizing your Octavia installation, see [installation of Octavia using Director](#).

**NOTE**

When leveraging Kuryr SDN, the Overcloud installation requires the Neutron **trunk** extension. This is available by default on director deployments. Use the **openvswitch** firewall instead of the default **ovs-hybrid** when the Neutron backend is ML2/OVS. There is no need for modifications if the backend is ML2/OVN.

6. In RHOSP versions earlier than 13.0.13, add the project ID to the **octavia.conf** configuration file after you create the project.
 - To enforce network policies across services, like when traffic goes through the Octavia load balancer, you must ensure Octavia creates the Amphora VM security groups on the user project. This change ensures that required load balancer security groups belong to that project, and that they can be updated to enforce services isolation.

**NOTE**

This task is unnecessary in RHOSP version 13.0.13 or later.

Octavia implements a new ACL API that restricts access to the load balancers VIP.

- a. Get the project ID

```
$ openstack project show <project>
```

Example output

```
+-----+-----+
| Field  | Value                |
+-----+-----+
| description |                    |
| domain_id | default              |
| enabled   | True                 |
| id       | PROJECT_ID          |
| is_domain | False                |
| name     | *<project>*         |
| parent_id | default              |
| tags    | []                   |
+-----+-----+
```

- b. Add the project ID to **octavia.conf** for the controllers.
 - i. Source the **stackrc** file:

```
$ source stackrc # Undercloud credentials
```

- ii. List the Openstack controllers:

```
$ openstack server list
```

Example output

```
+-----+-----+-----+-----+-----+
| ID              | Name      | Status | Networks |
| Image          | Flavor   |        |          |
+-----+-----+-----+-----+
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE | ctlplane=192.168.24.8 | overcloud-full | controller |
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0   | ACTIVE |           | overcloud-full | compute    |
+-----+-----+-----+-----+
|
```

- iii. SSH into the controller(s).

```
$ ssh heat-admin@192.168.24.8
```

- iv. Edit the **octavia.conf** file to add the project into the list of projects where Amphora security groups are on the user's account.

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

- c. Restart the Octavia worker so the new configuration loads.

```
controller-0$ sudo docker restart octavia_worker
```



NOTE

Depending on your RHOSP environment, Octavia might not support UDP listeners. If you use Kuryr SDN on RHOSP version 13.0.13 or earlier, UDP services are not supported. RHOSP version 16 or later support UDP.

8.2.3.3.1. The Octavia OVN Driver

Octavia supports multiple provider drivers through the Octavia API.

To see all available Octavia provider drivers, on a command line, enter:

■

```
$ openstack loadbalancer provider list
```

Example output

```
+-----+-----+
| name | description |
+-----+-----+
| amphora | The Octavia Amphora driver. |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn | Octavia OVN driver. |
+-----+-----+
```

Beginning with RHOSP version 16, the Octavia OVN provider driver (**ovn**) is supported on OpenShift Container Platform on RHOSP deployments.

ovn is an integration driver for the load balancing that Octavia and OVN provide. It supports basic load balancing capabilities, and is based on OpenFlow rules. The driver is automatically enabled in Octavia by Director on deployments that use OVN Neutron ML2.

The Amphora provider driver is the default driver. If **ovn** is enabled, however, Kuryr uses it.

If Kuryr uses **ovn** instead of Amphora, it offers the following benefits:

- Decreased resource requirements. Kuryr does not require a load balancer VM for each service.
- Reduced network latency.
- Increased service creation speed by using OpenFlow rules instead of a VM for each service.
- Distributed load balancing actions across all nodes instead of centralized on Amphora VMs.

You can [configure your cluster to use the Octavia OVN driver](#) after your RHOSP cloud is upgraded from version 13 to version 16.

8.2.3.4. Known limitations of installing with Kuryr

Using OpenShift Container Platform with Kuryr SDN has several known limitations.

RHOSP general limitations

OpenShift Container Platform with Kuryr SDN does not support **Service** objects with type **NodePort**.

RHOSP version limitations

Using OpenShift Container Platform with Kuryr SDN has several limitations that depend on the RHOSP version.

- RHOSP versions before 16 use the default Octavia load balancer driver (Amphora). This driver requires that one Amphora load balancer VM is deployed per OpenShift Container Platform service. Creating too many services can cause you to run out of resources. Deployments of later versions of RHOSP that have the OVN Octavia driver disabled also use the Amphora driver. They are subject to the same resource concerns as earlier versions of RHOSP.
- Octavia RHOSP versions before 13.0.13 do not support UDP listeners. Therefore, OpenShift Container Platform UDP services are not supported.

- Octavia RHOSP versions before 13.0.13 cannot listen to multiple protocols on the same port. Services that expose the same port to different protocols, like TCP and UDP, are not supported.

RHOSP environment limitations

There are limitations when using Kuryr SDN that depend on your deployment environment.

Because of Octavia's lack of support for the UDP protocol and multiple listeners, if the RHOSP version is earlier than 13.0.13, Kuryr forces pods to use TCP for DNS resolution.

In Go versions 1.12 and earlier, applications that are compiled with CGO support disabled use UDP only. In this case, the native Go resolver does not recognize the **use-vc** option in **resolv.conf**, which controls whether TCP is forced for DNS resolution. As a result, UDP is still used for DNS resolution, which fails.

To ensure that TCP forcing is allowed, compile applications either with the environment variable **CGO_ENABLED** set to **1**, i.e. **CGO_ENABLED=1**, or ensure that the variable is absent.

In Go versions 1.13 and later, TCP is used automatically if DNS resolution using UDP fails.



NOTE

musl-based containers, including Alpine-based containers, do not support the **use-vc** option.

RHOSP upgrade limitations

As a result of the RHOSP upgrade process, the Octavia API might be changed, and upgrades to the Amphora images that are used for load balancers might be required.

You can address API changes on an individual basis.

If the Amphora image is upgraded, the RHOSP operator can handle existing load balancer VMs in two ways:

- Upgrade each VM by triggering a [load balancer failover](#).
- Leave responsibility for upgrading the VMs to users.

If the operator takes the first option, there might be short downtimes during failovers.

If the operator takes the second option, the existing load balancers will not support upgraded Octavia API features, like UDP listeners. In this case, users must recreate their Services to use these features.



IMPORTANT

If OpenShift Container Platform detects a new Octavia version that supports UDP load balancing, it recreates the DNS service automatically. The service recreation ensures that the service default supports UDP load balancing.

The recreation causes the DNS service approximately one minute of downtime.

8.2.3.5. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

8.2.3.6. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

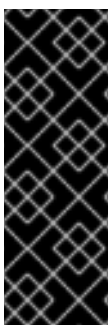
8.2.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

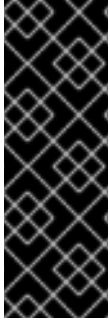


IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

8.2.5. Enabling Swift on RHOSP

Swift is operated by a user account with the **swiftoperator** role. Add the role to an account before you run the installation program.



IMPORTANT

If [the Red Hat OpenStack Platform \(RHOSP\) object storage service](#), commonly known as Swift, is available, OpenShift Container Platform uses it as the image registry storage. If it is unavailable, the installation program relies on the RHOSP block storage service, commonly known as Cinder.

If Swift is present and you want to use it, you must enable access to it. If it is not present, or if you do not want to use it, skip this section.

Prerequisites

- You have a RHOSP administrator account on the target environment.
- The Swift service is installed.
- On [Ceph RGW](#), the **account in url** option is enabled.

Procedure

To enable Swift on RHOSP:

1. As an administrator in the RHOSP CLI, add the **swiftoperator** role to the account that will access Swift:

```
$ openstack role add --user <user> --project <project> swiftoperator
```

Your RHOSP deployment can now use Swift for the image registry.

8.2.6. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the external router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries](#)

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

Example output

```
+-----+-----+-----+
```


ID	Name	Router Type
148a8023-62a7-4672-b018-003462f8d7dc	public_network	External

A network with an external router type appears in the network list. If at least one does not, see [Creating a default floating IP network](#) and [Creating a default provider network](#).

IMPORTANT

If the external network's CIDR range overlaps one of the default network ranges, you must change the matching network ranges in the **install-config.yaml** file before you start the installation process.

The default network ranges are:

Network	Range
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14



WARNING

If the installation program finds multiple networks with the same name, it sets one of them at random. To avoid this behavior, create unique names for resources in RHOSP.



NOTE

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

8.2.7. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:
 - If your RHOSP distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.



IMPORTANT

Remember to add a password to the **auth** field. You can also keep secrets in a [separate file](#) from **clouds.yaml**.

- If your RHOSP distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:

- a. Copy the certificate authority file to your machine.
- b. Add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

- c. Update the trust bundle:

```
$ sudo update-ca-trust extract
```

- d. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openshift/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openshift/clouds.yaml**

The installation program searches for **clouds.yaml** in that order.

8.2.8. Obtaining the installation program

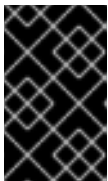
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

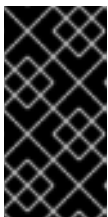
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

8.2.9. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

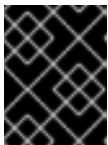


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.

- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Select **openstack** as the platform to target.
 - viii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - ix. Specify the floating IP address to use for external access to the OpenShift API.
 - x. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - xi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - xii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - xiii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

8.2.9.1. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

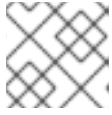
For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
  -----BEGIN CERTIFICATE-----
  <MY_TRUSTED_CA_CERT>
  -----END CERTIFICATE-----
...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

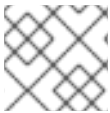
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

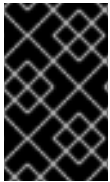
Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

8.2.10. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

8.2.10.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 8.9. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String


Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name> . <baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev . The string must be 14 characters or fewer long.
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


8.2.10.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 8.10. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .

Parameter	Description	Values
networking.serviceNetwork	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	<p>Required if you use networking.machineNetwork. An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24.</p>	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div>


8.2.10.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 8.11. Optional parameters

Parameter	Description	Values
additionalTrustBundle	<p>A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.</p>	String
compute	<p>The configuration for the machines that comprise the compute nodes.</p>	<p>Array of machine-pool objects. For details, see the following "Machine-pool" table.</p>

Parameter	Description	Values
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

8.2.10.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters

Additional RHOSP configuration parameters are described in the following table:

Table 8.12. Additional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .

8.2.10.5. Optional RHOSP configuration parameters

Optional RHOSP configuration parameters are described in the following table:

Table 8.13. Optional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.additionalNetworkIDs	Additional networks that are associated with compute machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
compute.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with compute machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
controlPlane.platform.openstack.additionalNetworkIDs	Additional networks that are associated with control plane machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
controlPlane.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with control plane machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
platform.openstack.clusterOSImage	<p>The location from which the installer downloads the RHCOS image.</p> <p>You must set this parameter to perform an installation in a restricted network.</p>	<p>An HTTP or HTTPS URL, optionally with an SHA-256 checksum.</p> <p>For example, http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d.</p> <p>The value can also be the name of an existing Glance image, for example my-rhcos.</p>
platform.openstack.defaultMachinePlatform	The default machine pool platform configuration.	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>
platform.openstack.externalDNS	IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings. For example, ["8.8.8.8", "192.168.1.12"] .

Parameter	Description	Values
platform.openstack.machinesSubnet	<p>The UUID of a RHOSP subnet that the cluster's nodes use. Nodes and virtual IP (VIP) ports are created on this subnet.</p> <p>The first item in networking.machineNetwork must match the value of machinesSubnet.</p> <p>If you deploy to a custom subnet, you cannot specify an external DNS server to the OpenShift Container Platform installer. Instead, add DNS to the subnet in RHOSP.</p>	A UUID as a string, for example fa806b2f-ac49-4bce-b9db-124bc64209bf .

8.2.10.6. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 8.14. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

8.2.10.7. Custom subnets in RHOSP deployments

Optionally, you can deploy a cluster on a Red Hat OpenStack Platform (RHOSP) subnet of your choice. The subnet's GUID is passed as the value of **platform.openstack.machinesSubnet** in the **install-config.yaml** file.

This subnet is used as the cluster's primary subnet; nodes and ports are created on it.

Before you run the OpenShift Container Platform installer with a custom subnet, verify that:

- The target network and subnet are available.
- DHCP is enabled on the target subnet.
- You can provide installer credentials that have permission to create ports on the target network.
- If your network configuration requires a router, it is created in RHOSP. Some configurations rely on routers for floating IP address translation.
- Your network configuration does not rely on a provider network. Provider networks are not supported.

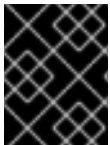


NOTE

By default, the API VIP takes x.x.x.5 and the Ingress VIP takes x.x.x.7 from your network's CIDR block. To override these default values, set values for **platform.openstack.apiVIP** and **platform.openstack.ingressVIP** that are outside of the DHCP allocation pool.

8.2.10.8. Sample customized install-config.yaml file for RHOSP with Kuryr

To deploy with Kuryr SDN instead of the default OpenShift SDN, you must modify the **install-config.yaml** file to include **Kuryr** as the desired **networking.networkType** and proceed with the default OpenShift Container Platform SDN installation steps. This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: m1.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
  hostPrefix: 23
  machineNetwork:
```

```

- cidr: 10.0.0.0/16
serviceNetwork:
- 172.30.0.0/16 1
networkType: Kuryr
platform:
openstack:
cloud: mycloud
externalNetwork: external
computeFlavor: m1.xlarge
lbFloatingIP: 128.0.0.1
trunkSupport: true 2
octaviaSupport: true 3
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...

```

- 1 The Amphora Octavia driver creates two ports per load balancer. As a result, the service subnet that the installer creates is twice the size of the CIDR that is specified as the value of the **serviceNetwork** property. The larger range is required to prevent IP address conflicts.
- 2 3 Both **trunkSupport** and **octaviaSupport** are automatically discovered by the installer, so there is no need to set them. But if your environment does not meet both requirements, Kuryr SDN will not properly work. Trunks are needed to connect the pods to the RHOSP network and Octavia is required to create the OpenShift Container Platform services.

8.2.11. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's **~/.ssh/authorized_keys** list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```

$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1

```

- 1 Specify the path and file name, such as **~/.ssh/id_rsa**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

8.2.12. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible with or without floating IP addresses.

8.2.12.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.



IMPORTANT

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>  
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```



NOTE

If you do not control the DNS server you can add the record to your **/etc/hosts** file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

8.2.12.2. Enabling access without floating IP addresses

If you cannot use floating IP addresses, the OpenShift Container Platform installation might still finish. However, the installation program fails after it times out waiting for API access.

After the installation program times out, the cluster might still initialize. After the bootstrapping processing begins, it must complete. You must edit the cluster's networking configuration after it is deployed.

8.2.13. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
 - The `~/.gcp/osServiceAccount.json` file
 - The **gcloud cli** default credentials
2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

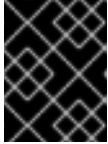
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrap** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

8.2.14. Verifying cluster status

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:

```
$ oc get clusterversion
```

4. View your Operators' status:

```
$ oc get clusteroperator
```

5. View all running pods in the cluster:

```
$ oc get pods -A
```

8.2.15. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

8.2.16. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

**NOTE**

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.2.17. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- If you need to enable external access to node ports, [configure ingress cluster traffic by using a node port.](#)

8.3. INSTALLING A CLUSTER ON OPENSTACK ON YOUR OWN INFRASTRUCTURE

In OpenShift Container Platform version 4.5, you can install a cluster on Red Hat OpenStack Platform (RHOSP) that runs on user-provisioned infrastructure.

Using your own infrastructure allows you to integrate your cluster with existing infrastructure and modifications. The process requires more labor on your part than installer-provisioned installations, because you must create all RHOSP resources, like Nova servers, Neutron ports, and security groups. However, Red Hat provides Ansible playbooks to help you in the deployment process.

8.3.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.5 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#) .
- Verify that your network configuration does not rely on a provider network. Provider networks are not supported.
- Have an RHOSP account where you want to install OpenShift Container Platform.
- On the machine from which you run the installation program, have:
 - A single directory in which you can keep the files you create during the installation process
 - Python 3

8.3.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

8.3.3. Resource guidelines for installing OpenShift Container Platform on RHOSP

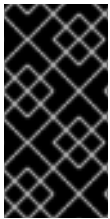
To support an OpenShift Container Platform installation, your Red Hat OpenStack Platform (RHOSP) quota must meet the following requirements:

Table 8.15. Recommended resources for a default OpenShift Container Platform cluster on RHOSP

Resource	Value
Floating IP addresses	3
Ports	15
Routers	1
Subnets	1
RAM	112 GB
vCPUs	28
Volume storage	275 GB

Resource	Value
Instances	7
Security groups	3
Security group rules	60

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If RHOSP object storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



NOTE

By default, your security group and security group rule quotas might be low. If you encounter problems, run **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** as an administrator to increase them.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

8.3.3.1. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

8.3.3.2. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

8.3.4. Downloading playbook dependencies

The Ansible playbooks that simplify the installation process on user-provisioned infrastructure require several Python modules. On the machine where you will run the installer, add the modules' repositories and then download them.



NOTE

These instructions assume that you are using Red Hat Enterprise Linux (RHEL) 8.

Prerequisites

- Python 3 is installed on your machine

Procedure

1. On a command line, add the repositories:

- a. Register with Red Hat Subscription Manager:

```
$ sudo subscription-manager register # If not done already
```

- b. Pull the latest subscription data:

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

- c. Disable the current repositories:

```
$ sudo subscription-manager repos --disable=* # If not done already
```

- d. Add the required repositories:

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. Install the modules:

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. Ensure that the **python** command points to **python3**:

```
$ sudo alternatives --set python /usr/bin/python3
```

8.3.5. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

8.3.6. Generating an SSH private key and adding it to the agent

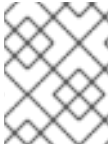
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

8.3.7. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image

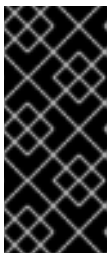
The OpenShift Container Platform installation program requires that a Red Hat Enterprise Linux CoreOS (RHCOS) image be present in the Red Hat OpenStack Platform (RHOSP) cluster. Retrieve the latest RHCOS image, then upload it using the RHOSP CLI.

Prerequisites

- The RHOSP CLI is installed.

Procedure

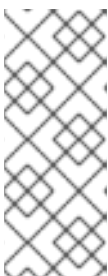
1. Log in to the Red Hat customer portal's [Product Downloads page](#).
2. Under **Version**, select the most recent release of OpenShift Container Platform 4.5 for Red Hat Enterprise Linux (RHEL) 8.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

3. Download the *Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)* .
4. Decompress the image.



NOTE

You must decompress the RHOSP image before the cluster can use it. The name of the downloaded file might not contain a compression extension, like **.gz** or **.tgz**. To find out if or how the file is compressed, in a command line, enter:

```
$ file <name_of_downloaded_file>
```

- From the image that you downloaded, create an image that is named **rhcos** in your cluster by using the RHOSP CLI:

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```



IMPORTANT

Depending on your RHOSP environment, you might be able to upload the image in either **.raw** or **.qcow2** formats. If you use Ceph, you must use the **.raw** format.



WARNING

If the installation program finds multiple images with the same name, it chooses one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

After you upload the image to RHOSP, it is usable in the installation process.

8.3.8. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the external router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries

Procedure

- Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

Example output

```
+-----+-----+-----+
| ID                | Name          | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+-----+-----+-----+
```

A network with an external router type appears in the network list. If at least one does not, see [Creating a default floating IP network](#) and [Creating a default provider network](#).

**NOTE**

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

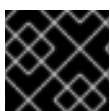
8.3.9. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible by using floating IP addresses.

8.3.9.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.

**IMPORTANT**

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```

**NOTE**

If you do not control the DNS server you can add the record to your **/etc/hosts** file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

8.3.10. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:

- If your RHOSP distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.



IMPORTANT

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your RHOSP distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:

- a. Copy the certificate authority file to your machine.
- b. Add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

- c. Update the trust bundle:

```
$ sudo update-ca-trust extract
```

- d. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
  ...
  cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openstack/clouds.yaml**

The installation program searches for **clouds.yaml** in that order.

8.3.11. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Select **openstack** as the platform to target.
 - viii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - ix. Specify the floating IP address to use for external access to the OpenShift API.
 - x. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - xi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - xii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - xiii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

You now have the file **install-config.yaml** in the directory that you specified.

8.3.12. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **install-config.yaml** file.

**IMPORTANT**

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

8.3.12.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 8.16. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .


Parameter	Description	Values
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev . The string must be 14 characters or fewer long.
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


8.3.12.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 8.17. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p>  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>



8.3.12.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 8.18. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: flex-start;"> <div style="flex: 1;">  </div> <div style="flex: 2;"> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA..

8.3.12.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters

Additional RHOSP configuration parameters are described in the following table:

Table 8.19. Additional RHOSP parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .

8.3.12.5. Optional RHOSP configuration parameters

Optional RHOSP configuration parameters are described in the following table:

Table 8.20. Optional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.additionalNetworkIDs	Additional networks that are associated with compute machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
compute.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with compute machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
controlPlane.platform.openstack.additionalNetworkIDs	Additional networks that are associated with control plane machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
controlPlane.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with control plane machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
platform.openstack.clusterOSImage	<p>The location from which the installer downloads the RHCOS image.</p> <p>You must set this parameter to perform an installation in a restricted network.</p>	<p>An HTTP or HTTPS URL, optionally with an SHA-256 checksum.</p> <p>For example, http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d.</p> <p>The value can also be the name of an existing Glance image, for example my-rhcos.</p>
platform.openstack.defaultMachinePlatform	The default machine pool platform configuration.	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>
platform.openstack.externalDNS	IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings. For example, ["8.8.8.8", "192.168.1.12"] .

Parameter	Description	Values
platform.openstack.machinesSubnet	<p>The UUID of a RHOSP subnet that the cluster's nodes use. Nodes and virtual IP (VIP) ports are created on this subnet.</p> <p>The first item in networking.machineNetwork must match the value of machinesSubnet.</p> <p>If you deploy to a custom subnet, you cannot specify an external DNS server to the OpenShift Container Platform installer. Instead, add DNS to the subnet in RHOSP.</p>	A UUID as a string, for example fa806b2f-ac49-4bce-b9db-124bc64209bf .

8.3.12.6. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 8.21. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

8.3.12.7. Custom subnets in RHOSP deployments

Optionally, you can deploy a cluster on a Red Hat OpenStack Platform (RHOSP) subnet of your choice. The subnet's GUID is passed as the value of **platform.openstack.machinesSubnet** in the **install-config.yaml** file.

This subnet is used as the cluster's primary subnet; nodes and ports are created on it.

Before you run the OpenShift Container Platform installer with a custom subnet, verify that:

- The target network and subnet are available.
- DHCP is enabled on the target subnet.
- You can provide installer credentials that have permission to create ports on the target network.
- If your network configuration requires a router, it is created in RHOSP. Some configurations rely on routers for floating IP address translation.
- Your network configuration does not rely on a provider network. Provider networks are not supported.

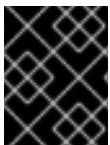


NOTE

By default, the API VIP takes x.x.x.5 and the Ingress VIP takes x.x.x.7 from your network's CIDR block. To override these default values, set values for **platform.openstack.apiVIP** and **platform.openstack.ingressVIP** that are outside of the DHCP allocation pool.

8.3.12.8. Sample customized install-config.yaml file for RHOSP

This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
```

```

hostPrefix: 23
machineNetwork:
- cidr: 10.0.0.0/16
serviceNetwork:
- 172.30.0.0/16
networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
  fips: false
  pullSecret: '{"auths": ...}'
  sshKey: ssh-ed25519 AAAA...

```

8.3.12.9. Setting a custom subnet for machines

The IP range that the installation program uses by default might not match the Neutron subnet that you create when you install OpenShift Container Platform. If necessary, update the CIDR value for new machines by editing the installation configuration file.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

- On a command line, browse to the directory that contains **install-config.yaml**.
- From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```

$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'

```

- 1** Insert a value that matches your intended Neutron subnet, e.g. **192.0.2.0/24**.

- To set the value manually, open the file and set the value of **networking.machineCIDR** to something that matches your intended Neutron subnet.

8.3.12.10. Emptying compute machine pools

To proceed with an installation that uses your own infrastructure, set the number of compute machines in the installation configuration file to zero. Later, you create these machines manually.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

- On a command line, browse to the directory that contains **install-config.yaml**.
- From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- To set the value manually, open the file and set the value of **compute.<first entry>.replicas** to **0**.

8.3.13. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

- Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

INFO Consuming Install Config from target directory
 WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings

- 1 For `<installation_directory>`, specify the installation directory that contains the `install-config.yaml` file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Remove the Kubernetes manifest files that define the control plane machines and compute machine sets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the machine set files to create compute machines by using the machine API, but you must update references to them to match your environment.
5. Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` file.
 - b. Locate the `mastersSchedulable` parameter and set its value to **False**.
 - c. Save and exit the file.
 6. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yaml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
```

```
baseDomain: example.openshift.com
privateZone: ❶
  id: mycluster-100419-private-zone
publicZone: ❷
  id: example.openshift.com
status: {}
```

❶ ❷ Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

7. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> ❶
```

❶ For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

8. Export the metadata file's **infraID** key as an environment variable:

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

TIP

Extract the **infraID** key from **metadata.json** and use it as a prefix for all of the RHOSP resources that you create. By doing so, you avoid name conflicts when making multiple deployments in the same project.

8.3.14. Preparing the bootstrap Ignition files

The OpenShift Container Platform installation process relies on bootstrap machines that are created from a bootstrap Ignition configuration file.

Edit the file and upload it. Then, create a secondary bootstrap Ignition configuration file that Red Hat OpenStack Platform (RHOSP) uses to download the primary file.

Prerequisites

- You have the bootstrap Ignition file that the installer program generates, **bootstrap.ign**.
- The infrastructure ID from the installer's metadata file is set as an environment variable (**\$INFRA_ID**).

- If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**
- You have an HTTP(S)-accessible way to store the bootstrap Ignition file.
 - The documented procedure uses the RHOSP image service (Glance), but you can also use the RHOSP storage service (Swift), Amazon S3, an internal HTTP server, or an ad hoc Nova server.

Procedure

1. Run the following Python script. The script modifies the bootstrap Ignition file to set the host name and, if available, CA certificate file when it runs:

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64,
        'verification': {}
    },
    'filesystem': 'root',
})

ca_cert_path = os.environ.get('OS_CACERT', "")
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64,
            'verification': {}
        },
        'filesystem': 'root',
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

- Using the RHOSP CLI, create an image that uses the bootstrap Ignition file:

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

- Get the image's details:

```
$ openstack image show <image_name>
```

Make a note of the **file** value; it follows the pattern **v2/images/<image_ID>/file**.



NOTE

Verify that the image you created is active.

- Retrieve the image service's public address:

```
$ openstack catalog show image
```

- Combine the public address with the image **file** value and save the result as the storage location. The location follows the pattern **<image_service_public_URL>/v2/images/<image_ID>/file**.

- Generate an auth token and save the token ID:

```
$ openstack token issue -c id -f value
```

- Insert the following content into a file called **\$INFRA_ID-bootstrap-ignition.json** and edit the placeholders to match your own values:

```
{
  "ignition": {
    "config": {
      "append": [{
        "source": "<storage_url>", 1
        "verification": {},
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>", 4
          "verification": {}
        }]
      }
    },
    "timeouts": {},
    "version": "2.4.0"
  },
}
```

```
"networkd": {},
"passwd": {},
"storage": {},
"systemd": {}
}
```

- 1 Replace the value of **ignition.config.append.source** with the bootstrap Ignition file storage URL.
- 2 Set **name** in **httpHeaders** to **"X-Auth-Token"**.
- 3 Set **value** in **httpHeaders** to your token's ID.
- 4 If the bootstrap Ignition file server uses a self-signed certificate, include the base64-encoded certificate.

8. Save the secondary Ignition config file.

The bootstrap Ignition data will be passed to RHOSP during installation.



WARNING

The bootstrap Ignition file contains sensitive information, like **clouds.yaml** credentials. Ensure that you store it in a secure place, and delete it after you complete the installation process.

8.3.15. Creating control plane Ignition config files

Installing OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) on your own infrastructure requires control plane Ignition config files. You must create multiple config files.



NOTE

As with the bootstrap Ignition configuration, you must explicitly define a host name for each control plane machine.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**

Procedure

- On a command line, run the following Python script:

```
$ for index in $(seq 0 2); do
  MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
  python -c "import base64, json, sys;
```

```

ignition = json.load(sys.stdin);
files = ignition['storage'].get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
ignition['storage']['files'] = files;
json.dump(ignition, sys.stdout) <master.ign >"$INFRA_ID-master-$index-ignition.json"
done

```

You now have three control plane Ignition files: **<INFRA_ID>-master-0-ignition.json**, **<INFRA_ID>-master-1-ignition.json**, and **<INFRA_ID>-master-2-ignition.json**.

8.3.16. Creating network resources

Create the network resources that an OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) installation on your own infrastructure requires. To save time, run supplied Ansible playbooks that generate security groups, networks, subnets, routers, and ports.

Procedure

1. Insert the following content into a local file that is called **common.yaml**:

Example 8.1. common.yaml Ansible playbook

```

- hosts: localhost
  gather_facts: no

  vars_files:
  - metadata.json

  tasks:
  - name: 'Compute resource names'
    set_fact:
      cluster_id_tag: "openshiftClusterID={{ infraID }}"
      os_network: "{{ infraID }}-network"
      os_subnet: "{{ infraID }}-nodes"
      os_router: "{{ infraID }}-external-router"
      # Port names
      os_port_api: "{{ infraID }}-api-port"
      os_port_ingress: "{{ infraID }}-ingress-port"
      os_port_bootstrap: "{{ infraID }}-bootstrap-port"
      os_port_master: "{{ infraID }}-master-port"
      os_port_worker: "{{ infraID }}-worker-port"
      # Security groups names
      os_sg_master: "{{ infraID }}-master"
      os_sg_worker: "{{ infraID }}-worker"
      # Server names
      os_bootstrap_server_name: "{{ infraID }}-bootstrap"
      os_cp_server_name: "{{ infraID }}-master"
      os_cp_server_group_name: "{{ infraID }}-master"
      os_compute_server_name: "{{ infraID }}-worker"
      # Trunk names
      os_cp_trunk_name: "{{ infraID }}-master-trunk"
      os_compute_trunk_name: "{{ infraID }}-worker-trunk"
      # Subnet pool name

```

```

subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
# Service network name
os_svc_network: "{{ infraID }}-kuryr-service-network"
# Service subnet name
os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
# Ignition files
os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"

```

2. Insert the following content into a local file that is called **inventory.yaml**:

Example 8.2. inventory.yaml Ansible playbook

```

all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'm1.xlarge'
      os_flavor_worker: 'm1.large'
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
      # OpenShift API floating IP address
      os_api_fip: '203.0.113.23'
      # OpenShift Ingress floating IP address
      os_ingress_fip: '203.0.113.19'
      # Service subnet cidr
      svc_subnet_range: '172.30.0.0/16'
      os_svc_network_range: '172.30.0.0/15'
      # Subnet pool prefixes
      cluster_network_cidrs: '10.128.0.0/14'
      # Subnet pool prefix length
      host_prefix: '23'
      # Name of the SDN.
      # Possible values are OpenshiftSDN or Kuryr.
      os_networking_type: 'OpenshiftSDN'

      # Number of provisioned Control Plane nodes
      # 3 is the minimum number for a fully-functional cluster.
      os_cp_nodes_number: 3

      # Number of provisioned Compute nodes.
      # 3 is the minimum number for a fully-functional cluster.
      os_compute_nodes_number: 3

```

3. Insert the following content into a local file that is called **security-groups.yaml**:

Example 8.3. security-groups.yaml

```

# Required Python packages:
#
# ansible

```

```
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Create the master security group'
      os_security_group:
        name: "{{ os_sg_master }}"

    - name: 'Set master security group tag'
      command:
        cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_master }}"

    - name: 'Create the worker security group'
      os_security_group:
        name: "{{ os_sg_worker }}"

    - name: 'Set worker security group tag'
      command:
        cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_worker }}"

    - name: 'Create master-sg rule "ICMP"'
      os_security_group_rule:
        security_group: "{{ os_sg_master }}"
        protocol: icmp

    - name: 'Create master-sg rule "machine config server"'
      os_security_group_rule:
        security_group: "{{ os_sg_master }}"
        protocol: tcp
        remote_ip_prefix: "{{ os_subnet_range }}"
        port_range_min: 22623
        port_range_max: 22623

    - name: 'Create master-sg rule "SSH"'
      os_security_group_rule:
        security_group: "{{ os_sg_master }}"
        protocol: tcp
        port_range_min: 22
        port_range_max: 22

    - name: 'Create master-sg rule "DNS (TCP)"'
      os_security_group_rule:
        security_group: "{{ os_sg_master }}"
        remote_ip_prefix: "{{ os_subnet_range }}"
        protocol: tcp
        port_range_min: 53
        port_range_max: 53

    - name: 'Create master-sg rule "DNS (UDP)"'
      os_security_group_rule:
        security_group: "{{ os_sg_master }}"
```



```
remote_ip_prefix: "{{ os_subnet_range }}"
protocol: udp
port_range_min: 53
port_range_max: 53

- name: 'Create master-sg rule "mDNS"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  remote_ip_prefix: "{{ os_subnet_range }}"
  protocol: udp
  port_range_min: 5353
  port_range_max: 5353

- name: 'Create master-sg rule "OpenShift API"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  port_range_min: 6443
  port_range_max: 6443

- name: 'Create master-sg rule "VXLAN"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 4789
  port_range_max: 4789

- name: 'Create master-sg rule "Geneve"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 6081
  port_range_max: 6081

- name: 'Create master-sg rule "ovndb"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 6641
  port_range_max: 6642

- name: 'Create master-sg rule "master ingress internal (TCP)"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal (UDP)"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
```

```
remote_ip_prefix: "{{ os_subnet_range }}"
port_range_min: 9000
port_range_max: 9999

- name: 'Create master-sg rule "kube scheduler"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube controller manager"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10257
    port_range_max: 10257

- name: 'Create master-sg rule "master ingress kubelet secure"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "etcd"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 2379
    port_range_max: 2380

- name: 'Create master-sg rule "master ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: '112'
```

```
remote_ip_prefix: "{{ os_subnet_range }}"

- name: 'Create worker-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: icmp

- name: 'Create worker-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create worker-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create worker-sg rule "Ingress HTTP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 80
    port_range_max: 80

- name: 'Create worker-sg rule "Ingress HTTPS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 443
    port_range_max: 443

- name: 'Create worker-sg rule "router"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 1936
    port_range_max: 1936

- name: 'Create worker-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create worker-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
```

```

protocol: udp
remote_ip_prefix: "{{ os_subnet_range }}"
port_range_min: 6081
port_range_max: 6081

- name: 'Create worker-sg rule "worker ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress kubelet insecure"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"

```

4. Insert the following content into a local file that is called **network.yaml**:

Example 8.4. network.yaml

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the cluster network'
    os_network:
      name: "{{ os_network }}"

  - name: 'Set the cluster network tag'
    command:
      cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_network }}"

  - name: 'Create a subnet'
    os_subnet:
      name: "{{ os_subnet }}"
      network_name: "{{ os_network }}"
      cidr: "{{ os_subnet_range }}"
      allocation_pool_start: "{{ os_subnet_range | next_nth_usable(10) }}"
      allocation_pool_end: "{{ os_subnet_range | ipaddr('last_usable') }}"

  - name: 'Set the cluster subnet tag'
    command:
      cmd: "openstack subnet set --tag {{ cluster_id_tag }} {{ os_subnet }}"

  - name: 'Create the service network'
    os_network:
      name: "{{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

  - name: 'Set the service network tag'
    command:
      cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

  - name: 'Computing facts for service subnet'
    set_fact:
      first_ip_svc_subnet_range: "{{ svc_subnet_range | ipv4('network') }}"
      last_ip_svc_subnet_range: "{{ svc_subnet_range | ipaddr('last_usable') | ipmath(1) }}"
      first_ip_os_svc_network_range: "{{ os_svc_network_range | ipv4('network') }}"
      last_ip_os_svc_network_range: "{{ os_svc_network_range | ipaddr('last_usable')
|ipmath(1) }}"
      allocation_pool: ""
    when: os_networking_type == "Kuryr"

  - name: 'Get first part of OpenStack network'
    set_fact:
      allocation_pool: "{{ allocation_pool + '--allocation-pool start={{

```

```

first_ip_os_svc_network_range | ipmath(1) }},end={{ first_ip_svc_subnet_range |ipmath(-
1) }}' }}"
  when:
  - os_networking_type == "Kuryr"
  - first_ip_svc_subnet_range != first_ip_os_svc_network_range

- name: 'Get last part of OpenStack network'
  set_fact:
    allocation_pool: "{{ allocation_pool + '--allocation-pool start={{
last_ip_svc_subnet_range | ipmath(1) }},end={{ last_ip_os_svc_network_range |ipmath(-
1) }}' }}"
  when:
  - os_networking_type == "Kuryr"
  - last_ip_svc_subnet_range != last_ip_os_svc_network_range

- name: 'Get end of allocation'
  set_fact:
    gateway_ip: "{{ allocation_pool.split('=')[-1] }}"
  when: os_networking_type == "Kuryr"

- name: 'replace last IP'
  set_fact:
    allocation_pool: "{{ allocation_pool | replace(gateway_ip, gateway_ip | ipmath(-1)) }}"
  when: os_networking_type == "Kuryr"

- name: 'list service subnet'
  command:
    cmd: "openstack subnet list --name {{ os_svc_subnet }} --tag {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: svc_subnet

- name: 'Create the service subnet'
  command:
    cmd: "openstack subnet create --ip-version 4 --gateway {{ gateway_ip }} --subnet-
range {{ os_svc_network_range }} {{ allocation_pool }} --no-dhcp --network {{
os_svc_network }} --tag {{ cluster_id_tag }} {{ os_svc_subnet }}"
  when:
  - os_networking_type == "Kuryr"
  - svc_subnet.stdout == ""

- name: 'list subnet pool'
  command:
    cmd: "openstack subnet pool list --name {{ subnet_pool }} --tags {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: pods_subnet_pool

- name: 'Create pods subnet pool'
  command:
    cmd: "openstack subnet pool create --default-prefix-length {{ host_prefix }} --pool-
prefix {{ cluster_network_cidrs }} --tag {{ cluster_id_tag }} {{ subnet_pool }}"
  when:
  - os_networking_type == "Kuryr"
  - pods_subnet_pool.stdout == ""

- name: 'Create external router'
  os_router:

```

```

name: "{{ os_router }}"
network: "{{ os_external_network }}"
interfaces:
- "{{ os_subnet }}"

- name: 'Set external router tag'
command:
  cmd: "openstack router set --tag {{ cluster_id_tag }} {{ os_router }}"
when: os_networking_type == "Kuryr"

- name: 'Create the API port'
os_port:
  name: "{{ os_port_api }}"
  network: "{{ os_network }}"
  security_groups:
  - "{{ os_sg_master }}"
  fixed_ips:
  - subnet: "{{ os_subnet }}"
    ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"

- name: 'Set API port tag'
command:
  cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_api }}"

- name: 'Create the Ingress port'
os_port:
  name: "{{ os_port_ingress }}"
  network: "{{ os_network }}"
  security_groups:
  - "{{ os_sg_worker }}"
  fixed_ips:
  - subnet: "{{ os_subnet }}"
    ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"

- name: 'Set the Ingress port tag'
command:
  cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_ingress }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the API floating IP to API port'
command:
  cmd: "openstack floating ip set --port {{ os_port_api }} {{ os_api_fip }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the Ingress floating IP to Ingress port'
command:
  cmd: "openstack floating ip set --port {{ os_port_ingress }} {{ os_ingress_fip }}"

```

5. On a command line, create security groups by running the **security-groups.yaml** playbook:

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

- On a command line, create a network, subnet, and router by running the **network.yaml** playbook:

```
$ ansible-playbook -i inventory.yaml network.yaml
```

- Optional: If you want to control the default resolvers that Nova servers use, run the RHOSP CLI command:

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

8.3.17. Creating the bootstrap machine

Create a bootstrap machine and give it the network access it needs to run on Red Hat OpenStack Platform (RHOSP). Red Hat provides an Ansible playbook that you run to simplify this process.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.json** file that the installation program created is in the same directory as the Ansible playbooks

Procedure

- On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
- Insert the following content into a local file that is called **bootstrap.yaml**:

Example 8.5. bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the bootstrap server port'
  os_port:
    name: "{{ os_port_bootstrap }}"
    network: "{{ os_network }}"
    security_groups:
    - "{{ os_sg_master }}"
    allowed_address_pairs:
    - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
```



```

- ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"

- name: 'Set bootstrap port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_bootstrap }}"

- name: 'Create the bootstrap server'
  os_server:
    name: "{{ os_bootstrap_server_name }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    userdata: "{{ lookup('file', os_bootstrap_ignition) | string }}"
    auto_ip: no
    nics:
      - port-name: "{{ os_port_bootstrap }}"

- name: 'Create the bootstrap floating IP'
  os_floating_ip:
    state: present
    network: "{{ os_external_network }}"
    server: "{{ os_bootstrap_server_name }}"

```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

4. After the bootstrap server is active, view the logs to verify that the Ignition files were received:

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

8.3.18. Creating the control plane machines

Create three control plane machines by using the Ignition config files that you generated.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The three Ignition files created in **Creating control plane Ignition config files**

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. If the control plane Ignition config files aren't already in your working directory, copy them into it.

3. Insert the following content into a local file that is called **control-plane.yaml**:

Example 8.6. control-plane.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
    - name: 'Create the Control Plane ports'
      os_port:
        name: "{{ item.1 }}-{{ item.0 }}"
        network: "{{ os_network }}"
        security_groups:
          - "{{ os_sg_master }}"
        allowed_address_pairs:
          - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
          - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
          - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
        with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
        register: ports

    - name: 'Set Control Plane ports tag'
      command:
        cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
        with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

    - name: 'List the Control Plane Trunks'
      command:
        cmd: "openstack network trunk list"
        when: os_networking_type == "Kuryr"
        register: control_plane_trunks

    - name: 'Create the Control Plane trunks'
      command:
        cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_cp_trunk_name }}-{{ item.0 }}"
        with_indexed_items: "{{ ports.results }}"
        when:
          - os_networking_type == "Kuryr"
          - "os_cp_trunk_name|string not in control_plane_trunks.stdout"

    - name: 'List the Server groups'
      command:
        cmd: "openstack server group list -f json -c ID -c Name"
        register: server_group_list

    - name: 'Parse the Server group ID from existing'
      set_fact:
```

```

server_group_id: "{{ (server_group_list.stdout | from_json | json_query(list_query) |
first).ID }}"
vars:
  list_query: "[?Name=='{{ os_cp_server_group_name }}]"
when:
  - "os_cp_server_group_name|string in server_group_list.stdout"

- name: 'Create the Control Plane server group'
  command:
    cmd: "openstack --os-compute-api-version=2.15 server group create -f json -c id --
policy=soft-anti-affinity {{ os_cp_server_group_name }}"
    register: server_group_created
  when:
    - server_group_id is not defined

- name: 'Parse the Server group ID from creation'
  set_fact:
    server_group_id: "{{ (server_group_created.stdout | from_json).id }}"
  when:
    - server_group_id is not defined

- name: 'Create the Control Plane servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    auto_ip: no
    # The ignition filename will be concatenated with the Control Plane node
    # name and its 0-indexed serial number.
    # In this case, the first node will look for this filename:
    # "{{ infraID }}-master-0-ignition.json"
    userdata: "{{ lookup('file', [item.1, item.0, 'ignition.json'] | join('-')) | string }}"
    nics:
      - port-name: "{{ os_port_master }}-{{ item.0 }}"
    scheduler_hints:
      group: "{{ server_group_id }}"
    with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

```

4. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

5. Run the following command to monitor the bootstrapping process:

```
$ openshift-install wait-for bootstrap-complete
```

You will see messages that confirm that the control plane machines are running and have joined the cluster:

```

INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources

```

8.3.19. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

8.3.20. Deleting bootstrap resources

Delete the bootstrap resources that you no longer need.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The control plane machines are running
 - If you don't know the machines' status, see **Verifying cluster status**

Procedure

1. Insert the following content into a local file that is called **down-bootstrap.yaml**:

Example 8.7. down-bootstrap.yaml

```
# Required Python packages:  
#  
# ansible  
# openstacksdk
```

```

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      state: absent
      delete_fip: yes

  - name: 'Remove the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      state: absent

```

2. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

The bootstrap port, server, and floating IP address are deleted.



WARNING

If you did not disable the bootstrap Ignition file URL earlier, do so now.

8.3.21. Creating compute machines

After standing up the control plane, create compute machines.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.json** file that the installation program created is in the same directory as the Ansible playbooks
- The control plane is active

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. Insert the following content into a local file that is called **compute-nodes.yaml**:

Example 8.8. compute-nodes.yaml

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the Compute ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
        - "{{ os_sg_worker }}"
      allowed_address_pairs:
        - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
    register: ports

  - name: 'Set Compute ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

  - name: 'List the Compute Trunks'
    command:
      cmd: "openstack network trunk list"
    when: os_networking_type == "Kuryr"
    register: compute_trunks

  - name: 'Create the Compute trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
      - os_networking_type == "Kuryr"
      - "os_compute_trunk_name|string not in compute_trunks.stdout"

  - name: 'Create the Compute servers'
    os_server:
      name: "{{ item.1 }}-{{ item.0 }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_worker }}"
      auto_ip: no
      userdata: "{{ lookup('file', 'worker.ign') | string }}"
      nics:
        - port-name: "{{ os_port_worker }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

Next steps

- Approve the machines' certificate signing requests

8.3.22. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:


```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{\n"}\n{{end}}\n' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

8.3.23. Verifying a successful installation

Verify that the OpenShift Container Platform installation is complete.

Prerequisites

- You have the installation program (**openshift-install**)

Procedure

- On a command line, enter:

```
$ openshift-install --log-level debug wait-for install-complete
```

The program outputs the console URL, as well as the administrator's login information.

8.3.24. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.3.25. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- If you need to enable external access to node ports, [configure ingress cluster traffic by using a node port.](#)

8.4. INSTALLING A CLUSTER ON OPENSTACK WITH KURYR ON YOUR OWN INFRASTRUCTURE

In OpenShift Container Platform version 4.5, you can install a cluster on Red Hat OpenStack Platform (RHOSP) that runs on user-provisioned infrastructure.

Using your own infrastructure allows you to integrate your cluster with existing infrastructure and modifications. The process requires more labor on your part than installer-provisioned installations, because you must create all RHOSP resources, like Nova servers, Neutron ports, and security groups. However, Red Hat provides Ansible playbooks to help you in the deployment process.

8.4.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
 - Verify that OpenShift Container Platform 4.5 is compatible with your RHOSP version in the *Available platforms* section. You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).
- Verify that your network configuration does not rely on a provider network. Provider networks are not supported.
- Have an RHOSP account where you want to install OpenShift Container Platform.
- On the machine from which you run the installation program, have:
 - A single directory in which you can keep the files you create during the installation process
 - Python 3

8.4.2. About Kuryr SDN

Kuryr is a container network interface (CNI) plug-in solution that uses the [Neutron](#) and [Octavia](#) Red Hat OpenStack Platform (RHOSP) services to provide networking for pods and Services.

Kuryr and OpenShift Container Platform integration is primarily designed for OpenShift Container Platform clusters running on RHOSP VMs. Kuryr improves the network performance by plugging OpenShift Container Platform pods into RHOSP SDN. In addition, it provides interconnectivity between pods and RHOSP virtual instances.

Kuryr components are installed as pods in OpenShift Container Platform using the **openshift-kuryr** namespace:

- **kuryr-controller** - a single service instance installed on a **master** node. This is modeled in OpenShift Container Platform as a **Deployment** object.
- **kuryr-cni** - a container installing and configuring Kuryr as a CNI driver on each OpenShift Container Platform node. This is modeled in OpenShift Container Platform as a **DaemonSet** object.

The Kuryr controller watches the OpenShift Container Platform API server for pod, service, and namespace create, update, and delete events. It maps the OpenShift Container Platform API calls to corresponding objects in Neutron and Octavia. This means that every network solution that implements the Neutron trunk port functionality can be used to back OpenShift Container Platform via Kuryr. This includes open source solutions such as Open vSwitch (OVS) and Open Virtual Network (OVN) as well as Neutron-compatible commercial SDNs.

Kuryr is recommended for OpenShift Container Platform deployments on encapsulated RHOSP tenant networks to avoid double encapsulation, such as running an encapsulated OpenShift Container Platform SDN over an RHOSP network.

If you use provider networks or tenant VLANs, you do not need to use Kuryr to avoid double encapsulation. The performance benefit is negligible. Depending on your configuration, though, using Kuryr to avoid having two overlays might still be beneficial.

Kuryr is not recommended in deployments where all of the following criteria are true:

- The RHOSP version is less than 16.
- The deployment uses UDP services, or a large number of TCP services on few hypervisors.

or

- The **ovn-octavia** Octavia driver is disabled.
- The deployment uses a large number of TCP services on few hypervisors.

8.4.3. Resource guidelines for installing OpenShift Container Platform on RHOSP with Kuryr

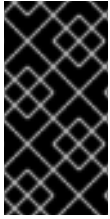
When using Kuryr SDN, the pods, services, namespaces, and network policies are using resources from the RHOSP quota; this increases the minimum requirements. Kuryr also has some additional requirements on top of what a default install requires.

Use the following quota to satisfy a default cluster's minimum requirements:

Table 8.22. Recommended resources for a default OpenShift Container Platform cluster on RHOSP with Kuryr

Resource	Value
Floating IP addresses	3 - plus the expected number of Services of LoadBalancer type
Ports	1500 - 1 needed per Pod
Routers	1
Subnets	250 - 1 needed per Namespace/Project
Networks	250 - 1 needed per Namespace/Project
RAM	112 GB
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	250 - 1 needed per Service and per NetworkPolicy
Security group rules	1000
Load balancers	100 - 1 needed per Service
Load balancer listeners	500 - 1 needed per Service-exposed port
Load balancer pools	500 - 1 needed per Service-exposed port

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If RHOSP object storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



IMPORTANT

If you are using Red Hat OpenStack Platform (RHOSP) version 16 with the Amphora driver rather than the OVN Octavia driver, security groups are associated with service accounts instead of user projects.

Take the following notes into consideration when setting resources:

- The number of ports that are required is larger than the number of pods. Kuryr uses ports pools to have pre-created ports ready to be used by pods and speed up the pods' booting time.
- Each network policy is mapped into an RHOSP security group, and depending on the **NetworkPolicy** spec, one or more rules are added to the security group.
- Each service is mapped to an RHOSP load balancer. Consider this requirement when estimating the number of security groups required for the quota.
If you are using RHOSP version 15 or earlier, or the **ovn-octavia driver**, each load balancer has a security group with the user project.
- The quota does not account for load balancer resources (such as VM resources), but you must consider these resources when you decide the RHOSP deployment's size. The default installation will have more than 50 load balancers; the clusters must be able to accommodate them.
If you are using RHOSP version 16 with the OVN Octavia driver enabled, only one load balancer VM is generated; services are load balanced through OVN flows.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

To enable Kuryr SDN, your environment must meet the following requirements:

- Run RHOSP 13+.
- Have Overcloud with Octavia.
- Use Neutron Trunk ports extension.
- Use **openvswitch** firewall driver if ML2/OVS Neutron driver is used instead of **ovs-hybrid**.

8.4.3.1. Increasing quota

When using Kuryr SDN, you must increase quotas to satisfy the Red Hat OpenStack Platform (RHOSP) resources used by pods, services, namespaces, and network policies.

Procedure

- Increase the quotas for a project by running the following command:

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets 250 --networks 250 <project>
```

8.4.3.2. Configuring Neutron

Kuryr CNI leverages the Neutron Trunks extension to plug containers into the Red Hat OpenStack Platform (RHOSP) SDN, so you must use the **trunks** extension for Kuryr to properly work.

In addition, if you leverage the default ML2/OVS Neutron driver, the firewall must be set to **openvswitch** instead of **ovs_hybrid** so that security groups are enforced on trunk subports and Kuryr can properly handle network policies.

8.4.3.3. Configuring Octavia

Kuryr SDN uses Red Hat OpenStack Platform (RHOSP)'s Octavia LBaaS to implement OpenShift Container Platform services. Thus, you must install and configure Octavia components in RHOSP to use Kuryr SDN.

To enable Octavia, you must include the Octavia service during the installation of the RHOSP Overcloud, or upgrade the Octavia service if the Overcloud already exists. The following steps for enabling Octavia apply to both a clean install of the Overcloud or an Overcloud update.



NOTE

The following steps only capture the key pieces required during the [deployment of RHOSP](#) when dealing with Octavia. It is also important to note that [registry methods](#) vary.

This example uses the local registry method.

Procedure

1. If you are using the local registry, create a template to upload the images to the registry. For example:

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{release} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. Verify that the **local_registry_images.yaml** file contains the Octavia images. For example:

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
```

```
push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
push_destination: <local-ip-from-undercloud.conf>:8787
```

**NOTE**

The Octavia container versions vary depending upon the specific RHOSP release installed.

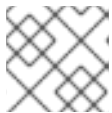
3. Pull the container images from **registry.redhat.io** to the Undercloud node:

```
(undercloud) $ sudo openstack overcloud container image upload \
--config-file /home/stack/local_registry_images.yaml \
--verbose
```

This may take some time depending on the speed of your network and Undercloud disk.

4. Since an Octavia load balancer is used to access the OpenShift Container Platform API, you must increase their listeners' default timeouts for the connections. The default timeout is 50 seconds. Increase the timeout to 20 minutes by passing the following file to the Overcloud deploy command:

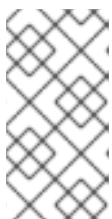
```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```

**NOTE**

This is not needed for RHOSP 13.0.13+.

5. Install or update your Overcloud environment with Octavia:

```
$ openstack overcloud deploy --templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
-e octavia_timeouts.yaml
```

**NOTE**

This command only includes the files associated with Octavia; it varies based on your specific installation of RHOSP. See the RHOSP documentation for further information. For more information on customizing your Octavia installation, see [installation of Octavia using Director](#).

**NOTE**

When leveraging Kuryr SDN, the Overcloud installation requires the Neutron **trunk** extension. This is available by default on director deployments. Use the **openvswitch** firewall instead of the default **ovs-hybrid** when the Neutron backend is ML2/OVS. There is no need for modifications if the backend is ML2/OVN.

6. In RHOSP versions earlier than 13.0.13, add the project ID to the **octavia.conf** configuration file after you create the project.
- To enforce network policies across services, like when traffic goes through the Octavia load balancer, you must ensure Octavia creates the Amphora VM security groups on the user project. This change ensures that required load balancer security groups belong to that project, and that they can be updated to enforce services isolation.



NOTE

This task is unnecessary in RHOSP version 13.0.13 or later.

Octavia implements a new ACL API that restricts access to the load balancers VIP.

- a. Get the project ID

```
$ openstack project show <project>
```

Example output

```
+-----+-----+
| Field  | Value                |
+-----+-----+
| description |                    |
| domain_id | default              |
| enabled   | True                 |
| id       | PROJECT_ID          |
| is_domain | False                |
| name     | *<project>*         |
| parent_id | default              |
| tags    | []                   |
+-----+-----+
```

- b. Add the project ID to **octavia.conf** for the controllers.

- i. Source the **stackrc** file:

```
$ source stackrc # Undercloud credentials
```

- ii. List the Overcloud controllers:

```
$ openstack server list
```

Example output

```
+-----+-----+-----+-----+-----+
| ID          | Name          | Status | Networks |
| Image      | Flavor       |        |          |
+-----+-----+-----+-----+-----+
```



```

+-----+-----+-----+-----+-----+
|
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
| ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0   | ACTIVE |
| ctlplane=192.168.24.6 | overcloud-full | compute  |
|
+-----+-----+-----+-----+-----+
|
+-----+

```

- iii. SSH into the controller(s).

```
$ ssh heat-admin@192.168.24.8
```

- iv. Edit the **octavia.conf** file to add the project into the list of projects where Amphora security groups are on the user's account.

```

# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID

```

- c. Restart the Octavia worker so the new configuration loads.

```
controller-0$ sudo docker restart octavia_worker
```



NOTE

Depending on your RHOSP environment, Octavia might not support UDP listeners. If you use Kuryr SDN on RHOSP version 13.0.13 or earlier, UDP services are not supported. RHOSP version 16 or later support UDP.

8.4.3.3.1. The Octavia OVN Driver

Octavia supports multiple provider drivers through the Octavia API.

To see all available Octavia provider drivers, on a command line, enter:

```
$ openstack loadbalancer provider list
```

Example output

```

+-----+-----+-----+-----+-----+
| name   | description                                     |
+-----+-----+-----+-----+-----+
| amphora | The Octavia Amphora driver.                   |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn    | Octavia OVN driver.                           |
+-----+-----+-----+-----+-----+

```

Beginning with RHOSP version 16, the Octavia OVN provider driver (**ovn**) is supported on OpenShift Container Platform on RHOSP deployments.

ovn is an integration driver for the load balancing that Octavia and OVN provide. It supports basic load balancing capabilities, and is based on OpenFlow rules. The driver is automatically enabled in Octavia by Director on deployments that use OVN Neutron ML2.

The Amphora provider driver is the default driver. If **ovn** is enabled, however, Kuryr uses it.

If Kuryr uses **ovn** instead of Amphora, it offers the following benefits:

- Decreased resource requirements. Kuryr does not require a load balancer VM for each service.
- Reduced network latency.
- Increased service creation speed by using OpenFlow rules instead of a VM for each service.
- Distributed load balancing actions across all nodes instead of centralized on Amphora VMs.

8.4.3.4. Known limitations of installing with Kuryr

Using OpenShift Container Platform with Kuryr SDN has several known limitations.

RHOSP general limitations

OpenShift Container Platform with Kuryr SDN does not support **Service** objects with type **NodePort**.

RHOSP version limitations

Using OpenShift Container Platform with Kuryr SDN has several limitations that depend on the RHOSP version.

- RHOSP versions before 16 use the default Octavia load balancer driver (Amphora). This driver requires that one Amphora load balancer VM is deployed per OpenShift Container Platform service. Creating too many services can cause you to run out of resources. Deployments of later versions of RHOSP that have the OVN Octavia driver disabled also use the Amphora driver. They are subject to the same resource concerns as earlier versions of RHOSP.
- Octavia RHOSP versions before 13.0.13 do not support UDP listeners. Therefore, OpenShift Container Platform UDP services are not supported.
- Octavia RHOSP versions before 13.0.13 cannot listen to multiple protocols on the same port. Services that expose the same port to different protocols, like TCP and UDP, are not supported.

RHOSP environment limitations

There are limitations when using Kuryr SDN that depend on your deployment environment.

Because of Octavia's lack of support for the UDP protocol and multiple listeners, if the RHOSP version is earlier than 13.0.13, Kuryr forces pods to use TCP for DNS resolution.

In Go versions 1.12 and earlier, applications that are compiled with CGO support disabled use UDP only. In this case, the native Go resolver does not recognize the **use-vc** option in **resolv.conf**, which controls whether TCP is forced for DNS resolution. As a result, UDP is still used for DNS resolution, which fails.

To ensure that TCP forcing is allowed, compile applications either with the environment variable **CGO_ENABLED** set to **1**, i.e. **CGO_ENABLED=1**, or ensure that the variable is absent.

In Go versions 1.13 and later, TCP is used automatically if DNS resolution using UDP fails.

**NOTE**

musl-based containers, including Alpine-based containers, do not support the **use-vc** option.

RHOSP upgrade limitations

As a result of the RHOSP upgrade process, the Octavia API might be changed, and upgrades to the Amphora images that are used for load balancers might be required.

You can address API changes on an individual basis.

If the Amphora image is upgraded, the RHOSP operator can handle existing load balancer VMs in two ways:

- Upgrade each VM by triggering a [load balancer failover](#).
- Leave responsibility for upgrading the VMs to users.

If the operator takes the first option, there might be short downtimes during failovers.

If the operator takes the second option, the existing load balancers will not support upgraded Octavia API features, like UDP listeners. In this case, users must recreate their Services to use these features.

**IMPORTANT**

If OpenShift Container Platform detects a new Octavia version that supports UDP load balancing, it recreates the DNS service automatically. The service recreation ensures that the service default supports UDP load balancing.

The recreation causes the DNS service approximately one minute of downtime.

8.4.3.5. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

8.4.3.6. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

8.4.4. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

8.4.5. Downloading playbook dependencies

The Ansible playbooks that simplify the installation process on user-provisioned infrastructure require several Python modules. On the machine where you will run the installer, add the modules' repositories and then download them.



NOTE

These instructions assume that you are using Red Hat Enterprise Linux (RHEL) 8.

Prerequisites

- Python 3 is installed on your machine

Procedure

1. On a command line, add the repositories:
 - a. Register with Red Hat Subscription Manager:

- a. Register with Red Hat Subscription Manager.

```
$ sudo subscription-manager register # If not done already
```

- b. Pull the latest subscription data:

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

- c. Disable the current repositories:

```
$ sudo subscription-manager repos --disable=* # If not done already
```

- d. Add the required repositories:

```
$ sudo subscription-manager repos \
--enable=rhel-8-for-x86_64-baseos-rpms \
--enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
--enable=ansible-2.9-for-rhel-8-x86_64-rpms \
--enable=rhel-8-for-x86_64-appstream-rpms
```

2. Install the modules:

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. Ensure that the **python** command points to **python3**:

```
$ sudo alternatives --set python /usr/bin/python3
```

8.4.6. Obtaining the installation program

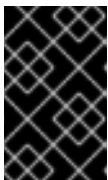
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

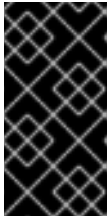
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

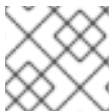
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

8.4.7. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

8.4.8. Creating the Red Hat Enterprise Linux CoreOS (RHCOS) image

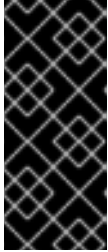
The OpenShift Container Platform installation program requires that a Red Hat Enterprise Linux CoreOS (RHCOS) image be present in the Red Hat OpenStack Platform (RHOSP) cluster. Retrieve the latest RHCOS image, then upload it using the RHOSP CLI.

Prerequisites

- The RHOSP CLI is installed.

Procedure

1. Log in to the Red Hat customer portal's [Product Downloads page](#).
2. Under **Version**, select the most recent release of OpenShift Container Platform 4.5 for Red Hat Enterprise Linux (RHEL) 8.

**IMPORTANT**

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

3. Download the *Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)* .
4. Decompress the image.

**NOTE**

You must decompress the RHOSP image before the cluster can use it. The name of the downloaded file might not contain a compression extension, like **.gz** or **.tgz**. To find out if or how the file is compressed, in a command line, enter:

```
$ file <name_of_downloaded_file>
```

5. From the image that you downloaded, create an image that is named **rhcos** in your cluster by using the RHOSP CLI:

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```

**IMPORTANT**

Depending on your RHOSP environment, you might be able to upload the image in either **.raw** or **.qcow2** formats. If you use Ceph, you must use the **.raw** format.

**WARNING**

If the installation program finds multiple images with the same name, it chooses one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

After you upload the image to RHOSP, it is usable in the installation process.

8.4.9. Verifying external network access

The OpenShift Container Platform installation process requires external network access. You must provide an external network value to it, or deployment fails. Before you begin the process, verify that a network with the external router type exists in Red Hat OpenStack Platform (RHOSP).

Prerequisites

- [Configure OpenStack's networking service to have DHCP agents forward instances' DNS queries](#)

Procedure

1. Using the RHOSP CLI, verify the name and ID of the 'External' network:

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

Example output

```
+-----+-----+-----+
| ID                | Name          | Router Type |
+-----+-----+-----+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+-----+-----+-----+
```

A network with an external router type appears in the network list. If at least one does not, see [Creating a default floating IP network](#) and [Creating a default provider network](#).



NOTE

If the Neutron trunk service plug-in is enabled, a trunk port is created by default. For more information, see [Neutron trunk port](#).

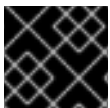
8.4.10. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible by using floating IP addresses.

8.4.10.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.



IMPORTANT

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>"
<external network>
```

- To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```



NOTE

If you do not control the DNS server you can add the record to your `/etc/hosts` file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

8.4.11. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

- Create the **clouds.yaml** file:
 - If your RHOSP distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.



IMPORTANT

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your RHOSP distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
    dev-env:
      region_name: RegionOne
      auth:
```

```
username: 'devuser'
password: XXX
project_name: 'devonly'
auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:

- a. Copy the certificate authority file to your machine.
- b. Add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

- c. Update the trust bundle:

```
$ sudo update-ca-trust extract
```

- d. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```
clouds:
  shiftstack:
    ...
  cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openstack/clouds.yaml**
 The installation program searches for **clouds.yaml** in that order.

8.4.12. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

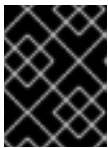


NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- v. Select the region to deploy the cluster to.
- vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- vii. Select **openstack** as the platform to target.
- viii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
- ix. Specify the floating IP address to use for external access to the OpenShift API.

- x. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - xi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - xii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - xiii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

You now have the file **install-config.yaml** in the directory that you specified.

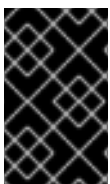
8.4.13. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

8.4.13.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 8.23. Required parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev . The string must be 14 characters or fewer long.
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object


Parameter	Description	Values
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

8.4.13.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 8.24. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>


Parameter	Description	Values
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.



8.4.13.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 8.25. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;"> <div style="background-color: black; width: 20px; height: 100px; margin-right: 10px;"></div> <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .

Parameter	Description	Values
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

8.4.13.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters

Additional RHOSP configuration parameters are described in the following table:

Table 8.26. Additional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .

8.4.13.5. Optional RHOSP configuration parameters

Optional RHOSP configuration parameters are described in the following table:

Table 8.27. Optional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.additionalNetworkIDs	Additional networks that are associated with compute machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
compute.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with compute machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
controlPlane.platform.openstack.additionalNetworkIDs	Additional networks that are associated with control plane machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
controlPlane.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with control plane machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
platform.openstack.clusterOSImage	<p>The location from which the installer downloads the RHCOS image.</p> <p>You must set this parameter to perform an installation in a restricted network.</p>	<p>An HTTP or HTTPS URL, optionally with an SHA-256 checksum.</p> <p>For example, http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d.</p> <p>The value can also be the name of an existing Glance image, for example my-rhcos.</p>
platform.openstack.defaultMachinePlatform	The default machine pool platform configuration.	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>
platform.openstack.externalDNS	IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings. For example, ["8.8.8.8", "192.168.1.12"] .

Parameter	Description	Values
platform.openstack.machinesSubnet	<p>The UUID of a RHOSP subnet that the cluster's nodes use. Nodes and virtual IP (VIP) ports are created on this subnet.</p> <p>The first item in networking.machineNetwork must match the value of machinesSubnet.</p> <p>If you deploy to a custom subnet, you cannot specify an external DNS server to the OpenShift Container Platform installer. Instead, add DNS to the subnet in RHOSP.</p>	A UUID as a string, for example fa806b2f-ac49-4bce-b9db-124bc64209bf .

8.4.13.6. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 8.28. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

8.4.13.7. Custom subnets in RHOSP deployments

Optionally, you can deploy a cluster on a Red Hat OpenStack Platform (RHOSP) subnet of your choice. The subnet's GUID is passed as the value of **platform.openstack.machinesSubnet** in the **install-config.yaml** file.

This subnet is used as the cluster's primary subnet; nodes and ports are created on it.

Before you run the OpenShift Container Platform installer with a custom subnet, verify that:

- The target network and subnet are available.
- DHCP is enabled on the target subnet.
- You can provide installer credentials that have permission to create ports on the target network.
- If your network configuration requires a router, it is created in RHOSP. Some configurations rely on routers for floating IP address translation.
- Your network configuration does not rely on a provider network. Provider networks are not supported.



NOTE

By default, the API VIP takes x.x.x.5 and the Ingress VIP takes x.x.x.7 from your network's CIDR block. To override these default values, set values for **platform.openstack.apiVIP** and **platform.openstack.ingressVIP** that are outside of the DHCP allocation pool.

8.4.13.8. Sample customized `install-config.yaml` file for RHOSP with Kuryr

To deploy with Kuryr SDN instead of the default OpenShift SDN, you must modify the **install-config.yaml** file to include **Kuryr** as the desired **networking.networkType** and proceed with the default OpenShift Container Platform SDN installation steps. This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.



IMPORTANT

This sample file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
```

```

- cidr: 10.0.0.0/16
serviceNetwork:
- 172.30.0.0/16 1
networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
    trunkSupport: true 2
    octaviaSupport: true 3
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...

```

- 1 The Amphora Octavia driver creates two ports per load balancer. As a result, the service subnet that the installer creates is twice the size of the CIDR that is specified as the value of the **serviceNetwork** property. The larger range is required to prevent IP address conflicts.
- 2 3 Both **trunkSupport** and **octaviaSupport** are automatically discovered by the installer, so there is no need to set them. But if your environment does not meet both requirements, Kuryr SDN will not properly work. Trunks are needed to connect the pods to the RHOSP network and Octavia is required to create the OpenShift Container Platform services.

8.4.13.9. Setting a custom subnet for machines

The IP range that the installation program uses by default might not match the Neutron subnet that you create when you install OpenShift Container Platform. If necessary, update the CIDR value for new machines by editing the installation configuration file.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

1. On a command line, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```

$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'

```

- 1 Insert a value that matches your intended Neutron subnet, e.g. **192.0.2.0/24**.

- To set the value manually, open the file and set the value of **networking.machineCIDR** to something that matches your intended Neutron subnet.

8.4.13.10. Emptying compute machine pools

To proceed with an installation that uses your own infrastructure, set the number of compute machines in the installation configuration file to zero. Later, you create these machines manually.

Prerequisites

- You have the **install-config.yaml** file that was generated by the OpenShift Container Platform installation program.

Procedure

1. On a command line, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- To set the value manually, open the file and set the value of **compute.<first entry>.replicas** to **0**.

8.4.13.11. Modifying the network type

By default, the installation program selects the **OpenShiftSDN** network type. To use Kuryr instead, change the value in the installation configuration file that the program generated.

Prerequisites

- You have the file **install-config.yaml** that was generated by the OpenShift Container Platform installation program

Procedure

1. In a command prompt, browse to the directory that contains **install-config.yaml**.
2. From that directory, either run a script to edit the **install-config.yaml** file or update the file manually:
 - To set the value by using a script, run:

```
$ python -c '
import yaml;
path = "install-config.yaml";
```

```
data = yaml.safe_load(open(path));
data["networking"]["networkType"] = "Kuryr";
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- To set the value manually, open the file and set **networking.networkType** to **"Kuryr"**.

8.4.14. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Remove the Kubernetes manifest files that define the control plane machines and compute machine sets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the machine set files to create compute machines by using the machine API, but you must update references to them to match your environment.
- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
 - Locate the `mastersSchedulable` parameter and set its value to **False**.
 - Save and exit the file.
 - Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- Remove this section completely.

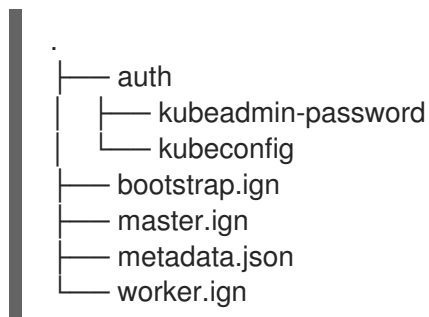
If you do so, you must add ingress DNS records manually in a later step.

- Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:



- Export the metadata file's **infraID** key as an environment variable:

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

TIP

Extract the **infraID** key from **metadata.json** and use it as a prefix for all of the RHOSP resources that you create. By doing so, you avoid name conflicts when making multiple deployments in the same project.

8.4.15. Preparing the bootstrap Ignition files

The OpenShift Container Platform installation process relies on bootstrap machines that are created from a bootstrap Ignition configuration file.

Edit the file and upload it. Then, create a secondary bootstrap Ignition configuration file that Red Hat OpenStack Platform (RHOSP) uses to download the primary file.

Prerequisites

- You have the bootstrap Ignition file that the installer program generates, **bootstrap.ign**.
- The infrastructure ID from the installer's metadata file is set as an environment variable (**\$INFRA_ID**).
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**
- You have an HTTP(S)-accessible way to store the bootstrap Ignition file.
 - The documented procedure uses the RHOSP image service (Glance), but you can also use the RHOSP storage service (Swift), Amazon S3, an internal HTTP server, or an ad hoc Nova server.

Procedure

- Run the following Python script. The script modifies the bootstrap Ignition file to set the host name and, if available, CA certificate file when it runs:

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
```

```

ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
    {
        'path': '/etc/hostname',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64,
            'verification': {}
        },
        'filesystem': 'root',
    })

ca_cert_path = os.environ.get('OS_CACERT', "")
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
        {
            'path': '/opt/openshift/tls/cloud-ca-cert.pem',
            'mode': 420,
            'contents': {
                'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64,
                'verification': {}
            },
            'filesystem': 'root',
        })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)

```

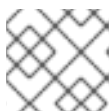
- Using the RHOSP CLI, create an image that uses the bootstrap Ignition file:

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

- Get the image's details:

```
$ openstack image show <image_name>
```

Make a note of the **file** value; it follows the pattern **v2/images/<image_ID>/file**.



NOTE

Verify that the image you created is active.

- Retrieve the image service's public address:

```
$ openstack catalog show image
```

- Combine the public address with the image **file** value and save the result as the storage location. The location follows the pattern **<image_service_public_URL>/v2/images/<image_ID>/file**.

- Generate an auth token and save the token ID:

```
$ openstack token issue -c id -f value
```

- Insert the following content into a file called **\$INFRA_ID-bootstrap-ignition.json** and edit the placeholders to match your own values:

```
{
  "ignition": {
    "config": {
      "append": [{
        "source": "<storage_url>", 1
        "verification": {},
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>", 4
          "verification": {}
        }]
      }
    },
    "timeouts": {},
    "version": "2.4.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

- Replace the value of **ignition.config.append.source** with the bootstrap Ignition file storage URL.
- Set **name** in **httpHeaders** to **"X-Auth-Token"**.
- Set **value** in **httpHeaders** to your token's ID.
- If the bootstrap Ignition file server uses a self-signed certificate, include the base64-encoded certificate.

8. Save the secondary Ignition config file.

The bootstrap Ignition data will be passed to RHOSP during installation.



WARNING

The bootstrap Ignition file contains sensitive information, like **clouds.yaml** credentials. Ensure that you store it in a secure place, and delete it after you complete the installation process.

8.4.16. Creating control plane Ignition config files

Installing OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) on your own infrastructure requires control plane Ignition config files. You must create multiple config files.



NOTE

As with the bootstrap Ignition configuration, you must explicitly define a host name for each control plane machine.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
 - If the variable is not set, see **Creating the Kubernetes manifest and Ignition config files**

Procedure

- On a command line, run the following Python script:

```
$ for index in $(seq 0 2); do
  MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
  python -c "import base64, json, sys;
  ignition = json.load(sys.stdin);
  files = ignition['storage'].get('files', []);
  files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
  ignition['storage']['files'] = files;
  json.dump(ignition, sys.stdout) <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

You now have three control plane Ignition files: **<INFRA_ID>-master-0-ignition.json**, **<INFRA_ID>-master-1-ignition.json**, and **<INFRA_ID>-master-2-ignition.json**.

8.4.17. Creating network resources

Create the network resources that an OpenShift Container Platform on Red Hat OpenStack Platform (RHOSP) installation on your own infrastructure requires. To save time, run supplied Ansible playbooks that generate security groups, networks, subnets, routers, and ports.

Procedure

1. Insert the following content into a local file that is called **common.yaml**:

Example 8.9. common.yaml Ansible playbook

```
- hosts: localhost
gather_facts: no

vars_files:
- metadata.json

tasks:
- name: 'Compute resource names'
  set_fact:
    cluster_id_tag: "openshiftClusterID={{ infraID }}"
    os_network: "{{ infraID }}-network"
    os_subnet: "{{ infraID }}-nodes"
    os_router: "{{ infraID }}-external-router"
    # Port names
    os_port_api: "{{ infraID }}-api-port"
    os_port_ingress: "{{ infraID }}-ingress-port"
    os_port_bootstrap: "{{ infraID }}-bootstrap-port"
    os_port_master: "{{ infraID }}-master-port"
    os_port_worker: "{{ infraID }}-worker-port"
    # Security groups names
    os_sg_master: "{{ infraID }}-master"
    os_sg_worker: "{{ infraID }}-worker"
    # Server names
    os_bootstrap_server_name: "{{ infraID }}-bootstrap"
    os_cp_server_name: "{{ infraID }}-master"
    os_cp_server_group_name: "{{ infraID }}-master"
    os_compute_server_name: "{{ infraID }}-worker"
    # Trunk names
    os_cp_trunk_name: "{{ infraID }}-master-trunk"
    os_compute_trunk_name: "{{ infraID }}-worker-trunk"
    # Subnet pool name
    subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
    # Service network name
    os_svc_network: "{{ infraID }}-kuryr-service-network"
    # Service subnet name
    os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
    # Ignition files
    os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. Insert the following content into a local file that is called **inventory.yaml**:

Example 8.10. inventory.yaml Ansible playbook

```
all:
hosts:
```



```

localhost:
  ansible_connection: local
  ansible_python_interpreter: "{{ansible_playbook_python}}"

# User-provided values
os_subnet_range: '10.0.0.0/16'
os_flavor_master: 'm1.xlarge'
os_flavor_worker: 'm1.large'
os_image_rhcos: 'rhcos'
os_external_network: 'external'
# OpenShift API floating IP address
os_api_fip: '203.0.113.23'
# OpenShift Ingress floating IP address
os_ingress_fip: '203.0.113.19'
# Service subnet cidr
svc_subnet_range: '172.30.0.0/16'
os_svc_network_range: '172.30.0.0/15'
# Subnet pool prefixes
cluster_network_cidrs: '10.128.0.0/14'
# Subnet pool prefix length
host_prefix: '23'
# Name of the SDN.
# Possible values are OpenshiftSDN or Kuryr.
os_networking_type: 'OpenshiftSDN'

# Number of provisioned Control Plane nodes
# 3 is the minimum number for a fully-functional cluster.
os_cp_nodes_number: 3

# Number of provisioned Compute nodes.
# 3 is the minimum number for a fully-functional cluster.
os_compute_nodes_number: 3

```

3. Insert the following content into a local file that is called **security-groups.yaml**:

Example 8.11. security-groups.yaml

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the master security group'
  os_security_group:
    name: "{{ os_sg_master }}"

- name: 'Set master security group tag'
  command:

```

```
cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_master }}"

- name: 'Create the worker security group'
  os_security_group:
    name: "{{ os_sg_worker }}"

- name: 'Set worker security group tag'
  command:
    cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_worker }}"

- name: 'Create master-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: icmp

- name: 'Create master-sg rule "machine config server"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 22623
    port_range_max: 22623

- name: 'Create master-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create master-sg rule "DNS (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: tcp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "DNS (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create master-sg rule "OpenShift API"'
  os_security_group_rule:
```

```
security_group: "{{ os_sg_master }}"
protocol: tcp
port_range_min: 6443
port_range_max: 6443

- name: 'Create master-sg rule "VXLAN"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 4789
  port_range_max: 4789

- name: 'Create master-sg rule "Geneve"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 6081
  port_range_max: 6081

- name: 'Create master-sg rule "ovndb"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 6641
  port_range_max: 6642

- name: 'Create master-sg rule "master ingress internal (TCP)"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal (UDP)"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 9000
  port_range_max: 9999

- name: 'Create master-sg rule "kube scheduler"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 10259
  port_range_max: 10259

- name: 'Create master-sg rule "kube controller manager"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
```

```
protocol: tcp
remote_ip_prefix: "{{ os_subnet_range }}"
port_range_min: 10257
port_range_max: 10257

- name: 'Create master-sg rule "master ingress kubelet secure"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 10250
  port_range_max: 10250

- name: 'Create master-sg rule "etcd"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 2379
  port_range_max: 2380

- name: 'Create master-sg rule "master ingress services (TCP)"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: tcp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 30000
  port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP)"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: udp
  remote_ip_prefix: "{{ os_subnet_range }}"
  port_range_min: 30000
  port_range_max: 32767

- name: 'Create master-sg rule "VRRP"'
os_security_group_rule:
  security_group: "{{ os_sg_master }}"
  protocol: '112'
  remote_ip_prefix: "{{ os_subnet_range }}"

- name: 'Create worker-sg rule "ICMP"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: icmp

- name: 'Create worker-sg rule "SSH"'
os_security_group_rule:
  security_group: "{{ os_sg_worker }}"
  protocol: tcp
  port_range_min: 22
  port_range_max: 22
```

- name: 'Create worker-sg rule "mDNS"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: udp
 remote_ip_prefix: "{{ os_subnet_range }}"
 port_range_min: 5353
 port_range_max: 5353

- name: 'Create worker-sg rule "Ingress HTTP"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: tcp
 port_range_min: 80
 port_range_max: 80

- name: 'Create worker-sg rule "Ingress HTTPS"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: tcp
 port_range_min: 443
 port_range_max: 443

- name: 'Create worker-sg rule "router"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: tcp
 remote_ip_prefix: "{{ os_subnet_range }}"
 port_range_min: 1936
 port_range_max: 1936

- name: 'Create worker-sg rule "VXLAN"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: udp
 remote_ip_prefix: "{{ os_subnet_range }}"
 port_range_min: 4789
 port_range_max: 4789

- name: 'Create worker-sg rule "Geneve"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: udp
 remote_ip_prefix: "{{ os_subnet_range }}"
 port_range_min: 6081
 port_range_max: 6081

- name: 'Create worker-sg rule "worker ingress internal (TCP)"'
os_security_group_rule:
 security_group: "{{ os_sg_worker }}"
 protocol: tcp
 remote_ip_prefix: "{{ os_subnet_range }}"
 port_range_min: 9000
 port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal (UDP)"'
os_security_group_rule:

```

security_group: "{{ os_sg_worker }}"
protocol: udp
remote_ip_prefix: "{{ os_subnet_range }}"
port_range_min: 9000
port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress kubelet insecure"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"

```

4. Insert the following content into a local file that is called **network.yaml**:

Example 8.12. network.yaml

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Create the cluster network'
  os_network:

```

```

    name: "{{ os_network }}"

- name: 'Set the cluster network tag'
  command:
    cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_network }}"

- name: 'Create a subnet'
  os_subnet:
    name: "{{ os_subnet }}"
    network_name: "{{ os_network }}"
    cidr: "{{ os_subnet_range }}"
    allocation_pool_start: "{{ os_subnet_range | next_nth_usable(10) }}"
    allocation_pool_end: "{{ os_subnet_range | ipaddr('last_usable') }}"

- name: 'Set the cluster subnet tag'
  command:
    cmd: "openstack subnet set --tag {{ cluster_id_tag }} {{ os_subnet }}"

- name: 'Create the service network'
  os_network:
    name: "{{ os_svc_network }}"
  when: os_networking_type == "Kuryr"

- name: 'Set the service network tag'
  command:
    cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_svc_network }}"
  when: os_networking_type == "Kuryr"

- name: 'Computing facts for service subnet'
  set_fact:
    first_ip_svc_subnet_range: "{{ svc_subnet_range | ipv4('network') }}"
    last_ip_svc_subnet_range: "{{ svc_subnet_range | ipaddr('last_usable') | ipmath(1) }}"
    first_ip_os_svc_network_range: "{{ os_svc_network_range | ipv4('network') }}"
    last_ip_os_svc_network_range: "{{ os_svc_network_range | ipaddr('last_usable')
|ipmath(1) }}"
    allocation_pool: ""
  when: os_networking_type == "Kuryr"

- name: 'Get first part of OpenStack network'
  set_fact:
    allocation_pool: "{{ allocation_pool + '--allocation-pool start={{
first_ip_os_svc_network_range | ipmath(1) }},end={{ first_ip_svc_subnet_range |ipmath(-
1) }}' }}"
  when:
    - os_networking_type == "Kuryr"
    - first_ip_svc_subnet_range != first_ip_os_svc_network_range

- name: 'Get last part of OpenStack network'
  set_fact:
    allocation_pool: "{{ allocation_pool + '--allocation-pool start={{
last_ip_svc_subnet_range | ipmath(1) }},end={{ last_ip_os_svc_network_range |ipmath(-
1) }}' }}"
  when:
    - os_networking_type == "Kuryr"
    - last_ip_svc_subnet_range != last_ip_os_svc_network_range

```

```
- name: 'Get end of allocation'
  set_fact:
    gateway_ip: "{{ allocation_pool.split('=')[-1] }}"
  when: os_networking_type == "Kuryr"

- name: 'replace last IP'
  set_fact:
    allocation_pool: "{{ allocation_pool | replace(gateway_ip, gateway_ip | ipmath(-1)) }}"
  when: os_networking_type == "Kuryr"

- name: 'list service subnet'
  command:
    cmd: "openstack subnet list --name {{ os_svc_subnet }} --tag {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: svc_subnet

- name: 'Create the service subnet'
  command:
    cmd: "openstack subnet create --ip-version 4 --gateway {{ gateway_ip }} --subnet-range {{ os_svc_network_range }} {{ allocation_pool }} --no-dhcp --network {{ os_svc_network }} --tag {{ cluster_id_tag }} {{ os_svc_subnet }}"
  when:
    - os_networking_type == "Kuryr"
    - svc_subnet.stdout == ""

- name: 'list subnet pool'
  command:
    cmd: "openstack subnet pool list --name {{ subnet_pool }} --tags {{ cluster_id_tag }}"
  when: os_networking_type == "Kuryr"
  register: pods_subnet_pool

- name: 'Create pods subnet pool'
  command:
    cmd: "openstack subnet pool create --default-prefix-length {{ host_prefix }} --pool-prefix {{ cluster_network_cidrs }} --tag {{ cluster_id_tag }} {{ subnet_pool }}"
  when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout == ""

- name: 'Create external router'
  os_router:
    name: "{{ os_router }}"
    network: "{{ os_external_network }}"
    interfaces:
      - "{{ os_subnet }}"

- name: 'Set external router tag'
  command:
    cmd: "openstack router set --tag {{ cluster_id_tag }} {{ os_router }}"
  when: os_networking_type == "Kuryr"

- name: 'Create the API port'
  os_port:
    name: "{{ os_port_api }}"
    network: "{{ os_network }}"
    security_groups:
```



```

- "{{ os_sg_master }}"
fixed_ips:
- subnet: "{{ os_subnet }}"
  ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"

- name: 'Set API port tag'
command:
  cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_api }}"

- name: 'Create the Ingress port'
os_port:
  name: "{{ os_port_ingress }}"
  network: "{{ os_network }}"
  security_groups:
  - "{{ os_sg_worker }}"
  fixed_ips:
  - subnet: "{{ os_subnet }}"
    ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"

- name: 'Set the Ingress port tag'
command:
  cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_ingress }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the API floating IP to API port'
command:
  cmd: "openstack floating ip set --port {{ os_port_api }} {{ os_api_fip }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the Ingress floating IP to Ingress port'
command:
  cmd: "openstack floating ip set --port {{ os_port_ingress }} {{ os_ingress_fip }}"

```

- On a command line, create security groups by running the **security-groups.yaml** playbook:

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

- On a command line, create a network, subnet, and router by running the **network.yaml** playbook:

```
$ ansible-playbook -i inventory.yaml network.yaml
```

- Optional: If you want to control the default resolvers that Nova servers use, run the RHOSP CLI command:

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

8.4.18. Creating the bootstrap machine

Create a bootstrap machine and give it the network access it needs to run on Red Hat OpenStack Platform (RHOSP). Red Hat provides an Ansible playbook that you run to simplify this process.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.json** file that the installation program created is in the same directory as the Ansible playbooks

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. Insert the following content into a local file that is called **bootstrap.yaml**:

Example 8.13. bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"

  - name: 'Set bootstrap port tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_bootstrap }}"

  - name: 'Create the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_master }}"
      userdata: "{{ lookup('file', os_bootstrap_ignition) | string }}"
      auto_ip: no
      nics:
      - port-name: "{{ os_port_bootstrap }}"
```

```
- name: 'Create the bootstrap floating IP'
  os_floating_ip:
    state: present
    network: "{{ os_external_network }}"
    server: "{{ os_bootstrap_server_name }}"
```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

4. After the bootstrap server is active, view the logs to verify that the Ignition files were received:

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

8.4.19. Creating the control plane machines

Create three control plane machines by using the Ignition config files that you generated.

Prerequisites

- The infrastructure ID from the installation program's metadata file is set as an environment variable (**\$INFRA_ID**)
- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The three Ignition files created in **Creating control plane Ignition config files**

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. If the control plane Ignition config files aren't already in your working directory, copy them into it.
3. Insert the following content into a local file that is called **control-plane.yaml**:

Example 8.14. control-plane.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no
```

```

tasks:
- name: 'Create the Control Plane ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_master }}"
    allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
    register: ports

- name: 'Set Control Plane ports tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

- name: 'List the Control Plane Trunks'
  command:
    cmd: "openstack network trunk list"
  when: os_networking_type == "Kuryr"
  register: control_plane_trunks

- name: 'Create the Control Plane trunks'
  command:
    cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_cp_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
      - os_networking_type == "Kuryr"
      - "os_cp_trunk_name|string not in control_plane_trunks.stdout"

- name: 'List the Server groups'
  command:
    cmd: "openstack server group list -f json -c ID -c Name"
    register: server_group_list

- name: 'Parse the Server group ID from existing'
  set_fact:
    server_group_id: "{{ (server_group_list.stdout | from_json | json_query(list_query) |
first).ID }}"
  vars:
    list_query: "[?Name=='{{ os_cp_server_group_name }}']"
  when:
    - "os_cp_server_group_name|string in server_group_list.stdout"

- name: 'Create the Control Plane server group'
  command:
    cmd: "openstack --os-compute-api-version=2.15 server group create -f json -c id --
policy=soft-anti-affinity {{ os_cp_server_group_name }}"
    register: server_group_created
  when:
    - server_group_id is not defined

```

```

- name: 'Parse the Server group ID from creation'
  set_fact:
    server_group_id: "{{ (server_group_created.stdout | from_json).id }}"
  when:
  - server_group_id is not defined

- name: 'Create the Control Plane servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    auto_ip: no
    # The ignition filename will be concatenated with the Control Plane node
    # name and its 0-indexed serial number.
    # In this case, the first node will look for this filename:
    # "{{ infraID }}-master-0-ignition.json"
    userdata: "{{ lookup('file', [item.1, item.0, 'ignition.json'] | join('-')) | string }}"
    nics:
      - port-name: "{{ os_port_master }}-{{ item.0 }}"
    scheduler_hints:
      group: "{{ server_group_id }}"
    with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

```

4. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

5. Run the following command to monitor the bootstrapping process:

```
$ openshift-install wait-for bootstrap-complete
```

You will see messages that confirm that the control plane machines are running and have joined the cluster:

```

INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources

```

8.4.20. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

8.4.21. Deleting bootstrap resources

Delete the bootstrap resources that you no longer need.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The control plane machines are running
 - If you don't know the machines' status, see **Verifying cluster status**

Procedure

1. Insert the following content into a local file that is called **down-bootstrap.yaml**:

Example 8.15. down-bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Remove the bootstrap server'
  os_server:
    name: "{{ os_bootstrap_server_name }}"
    state: absent
    delete_fip: yes
```

```
- name: 'Remove the bootstrap server port'
  os_port:
    name: "{{ os_port_bootstrap }}"
    state: absent
```

2. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

The bootstrap port, server, and floating IP address are deleted.



WARNING

If you did not disable the bootstrap Ignition file URL earlier, do so now.

8.4.22. Creating compute machines

After standing up the control plane, create compute machines.

Prerequisites

- The **inventory.yaml** and **common.yaml** Ansible playbooks in a common directory
 - If you need these files, copy them from **Creating network resources**
- The **metadata.json** file that the installation program created is in the same directory as the Ansible playbooks
- The control plane is active

Procedure

1. On a command line, change the working directory to the location of the **inventory.yaml** and **common.yaml** files.
2. Insert the following content into a local file that is called **compute-nodes.yaml**:

Example 8.16. compute-nodes.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
```

```

gather_facts: no

tasks:
- name: 'Create the Compute ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    network: "{{ os_network }}"
    security_groups:
      - "{{ os_sg_worker }}"
    allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
  register: ports

- name: 'Set Compute ports tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

- name: 'List the Compute Trunks'
  command:
    cmd: "openstack network trunk list"
  when: os_networking_type == "Kuryr"
  register: compute_trunks

- name: 'Create the Compute trunks'
  command:
    cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
  with_indexed_items: "{{ ports.results }}"
  when:
    - os_networking_type == "Kuryr"
    - "os_compute_trunk_name|string not in compute_trunks.stdout"

- name: 'Create the Compute servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_worker }}"
    auto_ip: no
    userdata: "{{ lookup('file', 'worker.ign') | string }}"
    nics:
      - port-name: "{{ os_port_worker }}-{{ item.0 }}"
  with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

```

3. On a command line, run the playbook:

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

Next steps

- Approve the machines' certificate signing requests

8.4.23. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   63m   v1.18.3
master-1  Ready   master   63m   v1.18.3
master-2  Ready   master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

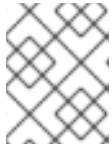
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

8.4.24. Verifying a successful installation

Verify that the OpenShift Container Platform installation is complete.

Prerequisites

- You have the installation program (**openshift-install**)

Procedure

- On a command line, enter:

```
$ openshift-install --log-level debug wait-for install-complete
```

The program outputs the console URL, as well as the administrator's login information.

8.4.25. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.4.26. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#).
- If you need to enable external access to node ports, [configure ingress cluster traffic by using a node port](#).

8.5. INSTALLING A CLUSTER ON OPENSTACK IN A RESTRICTED NETWORK

In OpenShift Container Platform 4.5, you can install a cluster on Red Hat OpenStack Platform (RHOSP) in a restricted network by creating an internal mirror of the installation release content.

Prerequisites

- [Create a registry on your mirror host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Review details about the [OpenShift Container Platform installation and update processes](#).
 - Verify that OpenShift Container Platform 4.5 is compatible with your RHOSP version by

consulting the architecture documentation’s [list of available platforms](#). You can also compare platform support across different versions by viewing the [OpenShift Container Platform on RHOSP support matrix](#).

- Verify that your network configuration does not rely on a provider network. Provider networks are not supported.
- Have the metadata service enabled in RHOSP.

8.5.1. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service’s IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.

8.5.1.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

8.5.2. Resource guidelines for installing OpenShift Container Platform on RHOSP

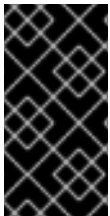
To support an OpenShift Container Platform installation, your Red Hat OpenStack Platform (RHOSP) quota must meet the following requirements:

Table 8.29. Recommended resources for a default OpenShift Container Platform cluster on RHOSP

Resource	Value
Floating IP addresses	3
Ports	15
Routers	1
Subnets	1
RAM	112 GB

Resource	Value
vCPUs	28
Volume storage	275 GB
Instances	7
Security groups	3
Security group rules	60

A cluster might function with fewer than recommended resources, but its performance is not guaranteed.



IMPORTANT

If RHOSP object storage (Swift) is available and operated by a user account with the **swiftoperator** role, it is used as the default backend for the OpenShift Container Platform image registry. In this case, the volume storage requirement is 175 GB. Swift space requirements vary depending on the size of the image registry.



NOTE

By default, your security group and security group rule quotas might be low. If you encounter problems, run **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** as an administrator to increase them.

An OpenShift Container Platform deployment comprises control plane machines, compute machines, and a bootstrap machine.

8.5.2.1. Control plane and compute machines

By default, the OpenShift Container Platform installation process stands up three control plane and three compute machines.

Each machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

TIP

Compute machines host the applications that you run on OpenShift Container Platform; aim to run as many as you can.

8.5.2.2. Bootstrap machine

During installation, a bootstrap machine is temporarily provisioned to stand up the control plane. After the production control plane is ready, the bootstrap machine is deprovisioned.

The bootstrap machine requires:

- An instance from the RHOSP quota
- A port from the RHOSP quota
- A flavor with at least 16 GB memory, 4 vCPUs, and 25 GB storage space

8.5.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

8.5.4. Enabling Swift on RHOSP

Swift is operated by a user account with the **swiftoperator** role. Add the role to an account before you run the installation program.



IMPORTANT

If the [Red Hat OpenStack Platform \(RHOSP\) object storage service](#), commonly known as Swift, is available, OpenShift Container Platform uses it as the image registry storage. If it is unavailable, the installation program relies on the RHOSP block storage service, commonly known as Cinder.

If Swift is present and you want to use it, you must enable access to it. If it is not present, or if you do not want to use it, skip this section.

Prerequisites

- You have a RHOSP administrator account on the target environment.
- The Swift service is installed.
- On [Ceph RGW](#), the **account in url** option is enabled.

Procedure

To enable Swift on RHOSP:

1. As an administrator in the RHOSP CLI, add the **swiftoperator** role to the account that will access Swift:

```
$ openstack role add --user <user> --project <project> swiftoperator
```

Your RHOSP deployment can now use Swift for the image registry.

8.5.5. Defining parameters for the installation program

The OpenShift Container Platform installation program relies on a file that is called **clouds.yaml**. The file describes Red Hat OpenStack Platform (RHOSP) configuration parameters, including the project name, log in information, and authorization service URLs.

Procedure

1. Create the **clouds.yaml** file:
 - If your RHOSP distribution includes the Horizon web UI, generate a **clouds.yaml** file in it.



IMPORTANT

Remember to add a password to the **auth** field. You can also keep secrets in [a separate file](#) from **clouds.yaml**.

- If your RHOSP distribution does not include the Horizon web UI, or you do not want to use Horizon, create the file yourself. For detailed information about **clouds.yaml**, see [Config files](#) in the RHOSP documentation.

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
```



```

username: shiftstack_user
password: XXX
user_domain_name: Default
project_domain_name: Default
dev-env:
region_name: RegionOne
auth:
  username: 'devuser'
  password: XXX
  project_name: 'devonly'
  auth_url: 'https://10.10.14.22:5001/v2.0'

```

2. If your RHOSP installation uses self-signed certificate authority (CA) certificates for endpoint authentication:

- a. Copy the certificate authority file to your machine.
- b. Add the machine to the certificate authority trust bundle:

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

- c. Update the trust bundle:

```
$ sudo update-ca-trust extract
```

- d. Add the **cacerts** key to the **clouds.yaml** file. The value must be an absolute, non-root-accessible path to the CA certificate:

```

clouds:
  shiftstack:
  ...
  cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"

```

TIP

After you run the installer with a custom CA certificate, you can update the certificate by editing the value of the **ca-cert.pem** key in the **cloud-provider-config** keymap. On a command line, run:

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. Place the **clouds.yaml** file in one of the following locations:
 - a. The value of the **OS_CLIENT_CONFIG_FILE** environment variable
 - b. The current directory
 - c. A Unix-specific user configuration directory, for example **~/.config/openstack/clouds.yaml**
 - d. A Unix-specific site configuration directory, for example **/etc/openstack/clouds.yaml**
The installation program searches for **clouds.yaml** in that order.

8.5.6. Creating the RHCOS image for restricted network installations

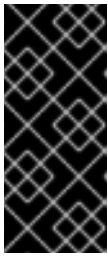
Download the Red Hat Enterprise Linux CoreOS (RHCOS) image to install OpenShift Container Platform on a restricted network Red Hat OpenStack Platform (RHOSP) environment.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, the program is on your mirror registry host.

Procedure

1. Log in to the Red Hat Customer Portal's [Product Downloads page](#).
2. Under **Version**, select the most recent release of OpenShift Container Platform 4.5 for RHEL 8.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

3. Download the **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)** image.
4. Decompress the image.



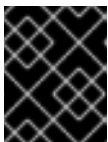
NOTE

You must decompress the image before the cluster can use it. The name of the downloaded file might not contain a compression extension, like **.gz** or **.tgz**. To find out if or how the file is compressed, in a command line, enter:

```
$ file <name_of_downloaded_file>
```

5. Upload the image that you decompressed to a location that is accessible from the bastion server, like Glance. For example:

```
$ openstack image create --file rhcos-44.81.202003110027-0-openstack.x86_64.qcow2 --disk-format qcow2 rhcos-${RHCOS_VERSION}
```



IMPORTANT

Depending on your RHOSP environment, you might be able to upload the image in either **.raw** or **.qcow2** formats. If you use Ceph, you must use the **.raw** format.

**WARNING**

If the installation program finds multiple images with the same name, it chooses one of them at random. To avoid this behavior, create unique names for resources in RHOSP.

The image is now available for a restricted installation. Note the image name or location for use in OpenShift Container Platform deployment.

8.5.7. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). Red Hat OpenStack Platform (RHOSP).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your bastion host.
- Have the **imageContentSources** values that were generated during mirror registry creation.
- Obtain the contents of the certificate for your mirror registry.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

**IMPORTANT**

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.

**NOTE**

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
 - iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - v. Select the region to deploy the cluster to.
 - vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
 - vii. Select **openstack** as the platform to target.
 - viii. Specify the Red Hat OpenStack Platform (RHOSP) external network name to use for installing the cluster.
 - ix. Specify the floating IP address to use for external access to the OpenShift API.
 - x. Specify a RHOSP flavor with at least 16 GB RAM to use for control plane and compute nodes.
 - xi. Select the base domain to deploy the cluster to. All DNS records will be sub-domains of this base and will also include the cluster name.
 - xii. Enter a name for your cluster. The name must be 14 or fewer characters long.
 - xiii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. In the **install-config.yaml** file, set the value of **platform.openstack.clusterOSImage** to the image location or name. For example:

```
platform:
  openstack:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
    openstack.x86_64.qcow2.gz?
    sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.
 - a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<bastion_host_name>:5000":{"auth": "<credentials>","email":
"you@example.com"}}}'
```

For **<bastion_host_name>**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

- b. Add the **additionalTrustBundle** parameter and value.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

  /-----END CERTIFICATE-----
```

The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

- c. Define the network and subnets for the VPC to install the cluster in under the parent **platform.gcp** field:

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

For **platform.gcp.network**, specify the name for the existing Google VPC. For **platform.gcp.controlPlaneSubnet** and **platform.gcp.computeSubnet**, specify the existing subnets to deploy the control plane machines and compute machines, respectively.

- d. Add the image content resources, which look like this excerpt:

```
imageContentSources:
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
    source: registry.example.com/ocp/release
```

To complete these values, use the **imageContentSources** that you recorded during mirror registry creation.

4. Make any other modifications to the **install-config.yaml** file that you require. You can find more information about the available parameters in the **Installation configuration parameters** section.
5. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

8.5.7.1. Installation configuration parameters

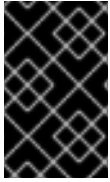
Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's

platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

8.5.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 8.30. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev . The string must be 14 characters or fewer long.

Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


8.5.7.1.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 8.31. Network parameters

Parameter	Description	Values
-----------	-------------	--------

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p>

8.5.7.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 8.32. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: center;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hypertreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div> </div>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

8.5.7.1.4. Additional Red Hat OpenStack Platform (RHOSP) configuration parameters

Additional RHOSP configuration parameters are described in the following table:

Table 8.33. Additional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.rootVolume.size	For compute machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
compute.platform.openstack.rootVolume.type	For compute machines, the root volume's type.	String, for example performance .
controlPlane.platform.openstack.rootVolume.size	For control plane machines, the size in gigabytes of the root volume. If you do not set this value, machines use ephemeral storage.	Integer, for example 30 .
controlPlane.platform.openstack.rootVolume.type	For control plane machines, the root volume's type.	String, for example performance .
platform.openstack.cloud	The name of the RHOSP cloud to use from the list of clouds in the clouds.yaml file.	String, for example MyCloud .
platform.openstack.externalNetwork	The RHOSP external network name to be used for installation.	String, for example external .
platform.openstack.computeFlavor	The RHOSP flavor to use for control plane and compute machines.	String, for example m1.xlarge .
platform.openstack.lbFloatingIP	An existing floating IP address to associate with the load balancer API.	An IP address, for example 128.0.0.1 .

8.5.7.1.5. Optional RHOSP configuration parameters

Optional RHOSP configuration parameters are described in the following table:

Table 8.34. Optional RHOSP parameters

Parameter	Description	Values
compute.platform.openstack.additionalNetworkIDs	Additional networks that are associated with compute machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
compute.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with compute machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
controlPlane.platform.openstack.additionalNetworkIDs	Additional networks that are associated with control plane machines. Allowed address pairs are not created for additional networks.	A list of one or more UUIDs as strings. For example, fa806b2f-ac49-4bce-b9db-124bc64209bf .
controlPlane.platform.openstack.additionalSecurityGroupIDs	Additional security groups that are associated with control plane machines.	A list of one or more UUIDs as strings. For example, 7ee219f3-d2e9-48a1-96c2-e7429f1b0da7 .
platform.openstack.clusterOSImage	<p>The location from which the installer downloads the RHCOS image.</p> <p>You must set this parameter to perform an installation in a restricted network.</p>	<p>An HTTP or HTTPS URL, optionally with an SHA-256 checksum.</p> <p>For example, http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d.</p> <p>The value can also be the name of an existing Glance image, for example my-rhcos.</p>
platform.openstack.defaultMachinePlatform	The default machine pool platform configuration.	<pre>{ "type": "ml.large", "rootVolume": { "size": 30, "type": "performance" } }</pre>
platform.openstack.externalDNS	IP addresses for external DNS servers that cluster instances use for DNS resolution.	A list of IP addresses as strings. For example, ["8.8.8.8", "192.168.1.12"] .

Parameter	Description	Values
platform.openstack.machinesSubnet	<p>The UUID of a RHOSP subnet that the cluster's nodes use. Nodes and virtual IP (VIP) ports are created on this subnet.</p> <p>The first item in networking.machineNetwork must match the value of machinesSubnet.</p> <p>If you deploy to a custom subnet, you cannot specify an external DNS server to the OpenShift Container Platform installer. Instead, add DNS to the subnet in RHOSP.</p>	A UUID as a string, for example fa806b2f-ac49-4bce-b9db-124bc64209bf .

8.5.7.1.6. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

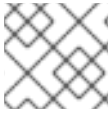
Table 8.35. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

8.5.7.2. Sample customized `install-config.yaml` file for restricted OpenStack installations

This sample **install-config.yaml** demonstrates all of the possible Red Hat OpenStack Platform (RHOSP) customization options.

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

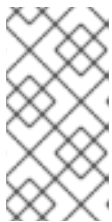
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

■

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

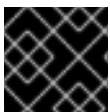
8.5.9. Enabling access to the environment

At deployment, all OpenShift Container Platform machines are created in a Red Hat OpenStack Platform (RHOSP)-tenant network. Therefore, they are not accessible directly in most RHOSP deployments.

You can configure the OpenShift Container Platform API and applications that run on the cluster to be accessible with or without floating IP addresses.

8.5.9.1. Enabling access with floating IP addresses

Create two floating IP (FIP) addresses: one for external access to the OpenShift Container Platform API, the **API FIP**, and one for OpenShift Container Platform applications, the **apps FIP**.

**IMPORTANT**

The API FIP is also used in the **install-config.yaml** file.

Procedure

1. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the API FIP:

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Using the Red Hat OpenStack Platform (RHOSP) CLI, create the apps, or Ingress, FIP:

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

- To reflect the new FIPs, add records that follow these patterns to your DNS server:

```
api.<cluster_name>.<base_domain>. IN A <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN A <apps_FIP>
```



NOTE

If you do not control the DNS server you can add the record to your `/etc/hosts` file instead. This action makes the API accessible to you only, which is not suitable for production deployment but does allow installation for development and testing.

TIP

You can make OpenShift Container Platform resources available outside of the cluster by assigning a floating IP address and updating your firewall configuration.

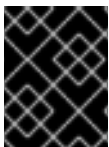
8.5.9.2. Enabling access without floating IP addresses

If you cannot use floating IP addresses, the OpenShift Container Platform installation might still finish. However, the installation program fails after it times out waiting for API access.

After the installation program times out, the cluster might still initialize. After the bootstrapping processing begins, it must complete. You must edit the cluster's networking configuration after it is deployed.

8.5.10. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Configure an account with the cloud platform that hosts your cluster.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

- Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCPLOUD_KEYFILE_JSON** environment variables
 - The `~/gcp/osServiceAccount.json` file
 - The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

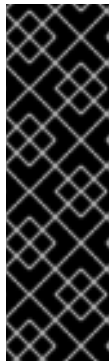
- 1 For **<installation_directory>**, specify the
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

8.5.11. Verifying cluster status

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:

```
$ oc get clusterversion
```

4. View your Operators' status:

```
$ oc get clusteroperator
```

5. View all running pods in the cluster:

```
$ oc get pods -A
```

8.5.12. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

8.5.13. Configuring application access with floating IP addresses

After you install OpenShift Container Platform, configure Red Hat OpenStack Platform (RHOSP) to allow application network traffic.

Prerequisites

- OpenShift Container Platform cluster must be installed
- Floating IP addresses are enabled as described in *Enabling access to the environment*.

Procedure

After you install the OpenShift Container Platform cluster, attach a floating IP address to the ingress port:

1. Show the port:

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. Attach the port to the IP address:

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. Add a wildcard **A** record for ***apps.** to your DNS file:

```
*.apps.<cluster name>.<base domain> IN A <apps FIP>
```

NOTE

If you do not control the DNS server but want to enable application access for non-production purposes, you can add these hostnames to **/etc/hosts**:

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oidc-server-openshift-authentication.apps.<cluster name>.<base domain>
<apps FIP> oidc-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

Next steps

- [Customize your cluster.](#)
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).
- Learn how to [use Operator Lifecycle Manager \(OLM\) on restricted networks](#).
- [Configure image streams](#) for the Cluster Samples Operator and the **must-gather** tool.

8.6. UNINSTALLING A CLUSTER ON OPENSTACK

You can remove a cluster that you deployed to Red Hat OpenStack Platform (RHOSP).

8.6.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access. For example, some Google Cloud resources require [IAM permissions](#) in shared VPC host projects, or there might be unused [health checks that must be deleted](#).

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

- 2** To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

8.7. UNINSTALLING A CLUSTER ON OPENSTACK FROM YOUR OWN INFRASTRUCTURE

You can remove a cluster that you deployed to Red Hat OpenStack Platform (RHOSP) on user-provisioned infrastructure.

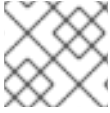
8.7.1. Prerequisites

- Have on your machine

- A single directory in which you can create files to help you with the removal process
- Python 3

8.7.2. Downloading playbook dependencies

The Ansible playbooks that simplify the removal process on user-provisioned infrastructure require several Python modules. On the machine where you will run the process, add the modules' repositories and then download them.



NOTE

These instructions assume that you are using Red Hat Enterprise Linux (RHEL) 8.

Prerequisites

- Python 3 is installed on your machine

Procedure

1. On a command line, add the repositories:

- a. Register with Red Hat Subscription Manager:

```
$ sudo subscription-manager register # If not done already
```

- b. Pull the latest subscription data:

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

- c. Disable the current repositories:

```
$ sudo subscription-manager repos --disable=* # If not done already
```

- d. Add the required repositories:

```
$ sudo subscription-manager repos \  
--enable=rhel-8-for-x86_64-baseos-rpms \  
--enable=openstack-16-tools-for-rhel-8-x86_64-rpms \  
--enable=ansible-2.9-for-rhel-8-x86_64-rpms \  
--enable=rhel-8-for-x86_64-appstream-rpms
```

2. Install the modules:

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk
```

3. Ensure that the **python** command points to **python3**:

```
$ sudo alternatives --set python /usr/bin/python3
```

8.7.3. Removing a cluster on RHOSP that uses your own infrastructure

You can remove an OpenShift Container Platform cluster on Red Hat OpenStack Platform (RHOSP) that uses your own infrastructure. To complete the removal process quickly, create and run several Ansible playbooks.

Prerequisites

- Python 3 is installed on your machine
- You downloaded the modules in "Downloading playbook dependencies"



PROCEDURE

You may have the **common.yaml** and **inventory.yaml** playbooks left over from when you installed OpenShift Container Platform. If you do, you can skip the first two steps of the procedure.

1. Insert the following content into a local file called **common.yaml**:

Example 8.17. common.yaml Ansible playbook

```
- hosts: localhost
gather_facts: no

vars_files:
- metadata.json

tasks:
- name: 'Compute resource names'
  set_fact:
    cluster_id_tag: "openshiftClusterID={{ infraID }}"
    os_network: "{{ infraID }}-network"
    os_subnet: "{{ infraID }}-nodes"
    os_router: "{{ infraID }}-external-router"
    # Port names
    os_port_api: "{{ infraID }}-api-port"
    os_port_ingress: "{{ infraID }}-ingress-port"
    os_port_bootstrap: "{{ infraID }}-bootstrap-port"
    os_port_master: "{{ infraID }}-master-port"
    os_port_worker: "{{ infraID }}-worker-port"
    # Security groups names
    os_sg_master: "{{ infraID }}-master"
    os_sg_worker: "{{ infraID }}-worker"
    # Server names
    os_bootstrap_server_name: "{{ infraID }}-bootstrap"
    os_cp_server_name: "{{ infraID }}-master"
    os_cp_server_group_name: "{{ infraID }}-master"
    os_compute_server_name: "{{ infraID }}-worker"
    # Trunk names
    os_cp_trunk_name: "{{ infraID }}-master-trunk"
    os_compute_trunk_name: "{{ infraID }}-worker-trunk"
    # Subnet pool name
    subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
    # Service network name
    os_svc_network: "{{ infraID }}-kuryr-service-network"
    # Service subnet name
```

```
os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
# Ignition files
os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. Insert the following content into a local file called **inventory.yaml**, and edit the values to match your own:

Example 8.18. inventory.yaml Ansible playbook

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'm1.xlarge'
      os_flavor_worker: 'm1.large'
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
      # OpenShift API floating IP address
      os_api_fip: '203.0.113.23'
      # OpenShift Ingress floating IP address
      os_ingress_fip: '203.0.113.19'
      # Service subnet cidr
      svc_subnet_range: '172.30.0.0/16'
      os_svc_network_range: '172.30.0.0/15'
      # Subnet pool prefixes
      cluster_network_cidrs: '10.128.0.0/14'
      # Subnet pool prefix length
      host_prefix: '23'
      # Name of the SDN.
      # Possible values are OpenshiftSDN or Kuryr.
      os_networking_type: 'OpenshiftSDN'

      # Number of provisioned Control Plane nodes
      # 3 is the minimum number for a fully-functional cluster.
      os_cp_nodes_number: 3

      # Number of provisioned Compute nodes.
      # 3 is the minimum number for a fully-functional cluster.
      os_compute_nodes_number: 3
```

3. *Optional:* If your cluster uses Kuryr, insert the following content into a local file called **down-load-balancers.yaml**:

Example 8.19. down-load-balancers.yaml

```
# Required Python packages:
#
# ansible
# openstackcli
# openstacksdk
```

```

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Get an auth token'
    os_auth:
    register: cloud
    when: os_networking_type == "Kuryr"

  - name: 'List octavia versions'
    uri:
    method: GET
    headers:
      X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
      Content-Type: 'application/json'
    url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
    json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url') }}"
    register: octavia_versions
    when: os_networking_type == "Kuryr"

  - set_fact:
    versions: "{{ octavia_versions.json.versions | selectattr('id', 'match', 'v2.5') |
    map(attribute='id') | list }}"
    when: os_networking_type == "Kuryr"

  - name: 'List tagged loadbalancers'
    uri:
    method: GET
    headers:
      X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
    url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
    json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url')
    }}/v2.0/lbaas/loadbalancers?tags={{cluster_id_tag}}"
    when:
    - os_networking_type == "Kuryr"
    - versions | length > 0
    register: lbs_tagged

    # NOTE: Kuryr creates an Octavia load balancer
    # for each service present on the cluster. Let's make
    # sure to remove the resources generated.
  - name: 'Remove the cluster load balancers'
    command:
    cmd: "openstack loadbalancer delete --cascade {{ item.id }}"
    with_items: "{{ lbs_tagged.json.loadbalancers }}"
    when:
    - os_networking_type == "Kuryr"
    - versions | length > 0
    - "'PENDING' not in item.provisioning_status"

  - name: 'List loadbalancers tagged on description'
    uri:
    method: GET

```

```

headers:
  X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
  url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url')
}}/v2.0/lbaas/loadbalancers?description={{cluster_id_tag}}"
  when:
    - os_networking_type == "Kuryr"
    - versions | length == 0
  register: lbs_description

# NOTE: Kuryr creates an Octavia load balancer
# for each service present on the cluster. Let's make
# sure to remove the resources generated.
- name: 'Remove the cluster load balancers'
  command:
    cmd: "openstack loadbalancer delete --cascade {{ item.id }}"
  with_items: "{{ lbs_description.json.loadbalancers }}"
  when:
    - os_networking_type == "Kuryr"
    - versions | length == 0
    - "PENDING" not in item.provisioning_status'

```

4. Insert the following content into a local file called **down-compute-nodes.yaml**:

Example 8.20. down-compute-nodes.yaml

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'Remove the Compute servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    state: absent
  with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

- name: 'List the Compute trunks'
  command:
    cmd: "openstack network trunk list -c Name -f value"
  when: os_networking_type == "Kuryr"
  register: trunks

- name: 'Remove the Compute trunks'
  command:
    cmd: "openstack network trunk delete {{ item.1 }}-{{ item.0 }}"
  when:
    - os_networking_type == "Kuryr"

```

```

- (item.1|string + '-' + item.0|string) in trunks.stdout_lines|list
with_indexed_items: "{{ [os_compute_trunk_name] * os_compute_nodes_number }}"

- name: 'Remove the Compute ports'
os_port:
  name: "{{ item.1 }}-{{ item.0 }}"
  state: absent
with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

```

5. Insert the following content into a local file called **down-control-plane.yaml**:

Example 8.21. down-control-plane.yaml

```

# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
gather_facts: no

tasks:
- name: 'Remove the Control Plane servers'
os_server:
  name: "{{ item.1 }}-{{ item.0 }}"
  state: absent
with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

- name: 'Remove the Control Plane server group'
os_server_group:
  name: "{{ os_cp_server_group_name }}"
  state: absent

- name: 'List the Compute trunks'
command:
  cmd: "openstack network trunk list -c Name -f value"
when: os_networking_type == "Kuryr"
register: trunks

- name: 'Remove the Control Plane trunks'
command:
  cmd: "openstack network trunk delete {{ item.1 }}-{{ item.0 }}"
when:
- os_networking_type == "Kuryr"
- (item.1|string + '-' + item.0|string) in trunks.stdout_lines|list
with_indexed_items: "{{ [os_cp_trunk_name] * os_cp_nodes_number }}"

- name: 'Remove the Control Plane ports'
os_port:
  name: "{{ item.1 }}-{{ item.0 }}"
  state: absent
with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

```

6. Insert the following content into a local file called **down-bootstrap.yaml**:

Example 8.22. down-bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      state: absent
      delete_fip: yes

  - name: 'Remove the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      state: absent
```

7. Insert the following content into a local file called **down-network.yaml**:

Example 8.23. down-network.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'List ports attached to router'
    command:
      cmd: "openstack port list --device-owner=network:router_interface --tags {{
cluster_id_tag }} -f value -c id"
    register: router_ports

  - name: 'Remove the ports from router'
    command:
      cmd: "openstack router remove port {{ os_router }} {{ item.1 }}"
    with_indexed_items: "{{ router_ports.stdout_lines }}"
```

```

- name: 'List ha ports attached to router'
  command:
    cmd: "openstack port list --device-owner=network:ha_router_replicated_interface --
tags {{ cluster_id_tag }} -f value -c id"
  register: ha_router_ports

- name: 'Remove the ha ports from router'
  command:
    cmd: "openstack router remove port {{ os_router }} {{ item.1 }}"
  with_indexed_items: "{{ ha_router_ports.stdout_lines }}"

- name: 'List ports'
  command:
    cmd: "openstack port list --tags {{ cluster_id_tag }} -f value -c id "
  register: ports

- name: 'Remove the cluster ports'
  command:
    cmd: "openstack port delete {{ item.1 }}"
  with_indexed_items: "{{ ports.stdout_lines }}"

- name: 'Remove the cluster router'
  os_router:
    name: "{{ os_router }}"
    state: absent

- name: 'List cluster networks'
  command:
    cmd: "openstack network list --tags {{ cluster_id_tag }} -f value -c Name"
  register: networks

- name: 'Remove the cluster networks'
  os_network:
    name: "{{ item.1 }}"
    state: absent
  with_indexed_items: "{{ networks.stdout_lines }}"

- name: 'List the cluster subnet pool'
  command:
    cmd: "openstack subnet pool list --name {{ subnet_pool }}"
  when: os_networking_type == "Kuryr"
  register: pods_subnet_pool

- name: 'Remove the cluster subnet pool'
  command:
    cmd: "openstack subnet pool delete {{ subnet_pool }}"
  when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout != ""

```

8. Insert the following content into a local file called **down-security-groups.yaml**:

Example 8.24. down-security-groups.yaml

```
# Required Python packages:
```

```
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

tasks:
- name: 'List security groups'
  command:
    cmd: "openstack security group list --tags {{ cluster_id_tag }} -f value -c ID"
  register: security_groups

- name: 'Remove the cluster security groups'
  command:
    cmd: "openstack security group delete {{ item.1 }}"
  with_indexed_items: "{{ security_groups.stdout_lines }}"
```

9. On a command line, run the playbooks you created:

```
$ ansible-playbook -i inventory.yaml \
down-bootstrap.yaml \
down-control-plane.yaml \
down-compute-nodes.yaml \
down-load-balancers.yaml \
down-network.yaml \
down-security-groups.yaml
```

10. Remove any DNS record changes you made for the OpenShift Container Platform installation.

OpenShift Container Platform is removed from your infrastructure.

CHAPTER 9. INSTALLING ON RHV

9.1. INSTALLING A CLUSTER QUICKLY ON RHV



WARNING

Due to a known issue, this default installation procedure does not work with OpenShift Container Platform versions 4.4 and 4.5 on Red Hat Virtualization (RHV) 4.4.1. This defect is fixed in RHV 4.4.2.

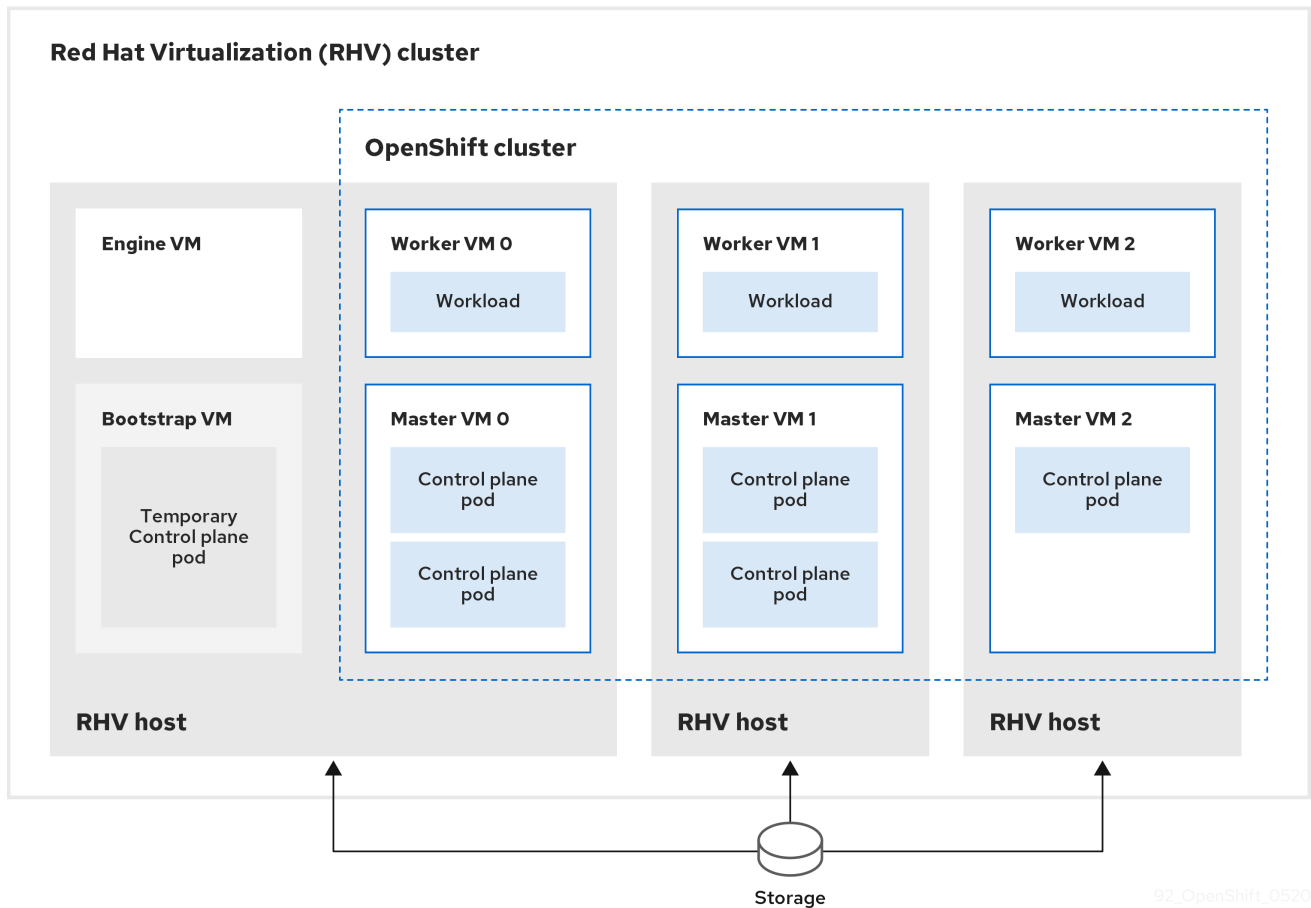
Instead, follow the steps in [Installing a cluster on RHV with customizations](#).



WARNING

Installing OpenShift Container Platform version 4.6 on Red Hat Virtualization (RHV) requires RHV version 4.4. If you are running an earlier version of OpenShift Container Platform on RHV 4.3, do not update it to OpenShift Container Platform version 4.6. Red Hat has not tested running OpenShift Container Platform version 4.6 on RHV version 4.3 and does not support this combination. For more information, see [OpenShift Container Platform 4.x Tested Integrations \(for x86_x64\)](#).

You can quickly install a default, non-customized, OpenShift Container Platform cluster on a Red Hat Virtualization (RHV) cluster, similar to the one shown in the following diagram.



92_OpenShift_0520

The installation program uses installer-provisioned infrastructure to automate creating and deploying the cluster.

To install a default cluster, you prepare the environment, run the installation program and answer its prompts. Then, the installation program creates the OpenShift Container Platform cluster.

For an alternative to installing a default cluster, see [Installing a cluster with customizations](#).



NOTE

This installation program is available for Linux and macOS only.

9.1.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, [configure it to allow the sites](#) that your cluster requires access to.

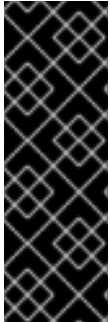
9.1.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

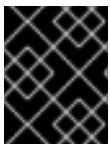
If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

9.1.3. Requirements for the RHV environment

To install and run an OpenShift Container Platform cluster, the RHV environment must meet the following requirements. Not meeting these requirements can cause failures.

The following requirements for CPUs, memory, and storage are based on *default* values multiplied by the default number of virtual machines the installation program creates.

By default, the installation program creates seven machines during the installation process, which includes one bootstrap machine. When the installation program finishes, it deletes the bootstrap machine and frees up its resources. If you perform a custom installation, you can increase the number of virtual machines the installation program creates.



IMPORTANT

If you increase the number of virtual machines or resources in the `install_config.yaml` file, you must also increase these requirements.

Requirements

- The RHV version is 4.3.10 or later.
- The RHV environment has one data center whose state is **Up**.
- The RHV data center contains an RHV cluster.
- The RHV cluster has the following resources exclusively for the OpenShift Container Platform cluster:
 - Minimum 28 vCPUs, which is 4 vCPUs for each of the seven virtual machines created during installation.
 - 112 GiB RAM or more, including:
 - 16 GiB or more for the bootstrap machine, which provides the temporary control plane.

- 16 GiB or more for each of the three control plane machines which provide the control plane.
- 16 GiB or more for each of the three compute machines, which run the application workloads.
- The RHV storage domain must meet [these etcd backend performance requirements](#).
- In production environments, each virtual machine must have 120 GiB or more, so the storage domain must have 840 GiB or more for the OpenShift Container Platform cluster. In resource-constrained or non-production environments, each virtual machine must have 32 GiB or more, so the storage domain must have 230 GiB or more for the OpenShift Container Platform cluster.
- The RHV cluster must have access to an Internet connection to download images from the Red Hat Ecosystem Catalog during installation and updates, and for the Telemetry service to simplify the subscription and entitlement process.
- The RHV cluster has a virtual network with access to the REST API on the RHV Manager. Ensure that DHCP is enabled on this network, because the VMs that the installer creates obtain their IP address by using DHCP.

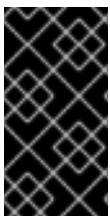


NOTE

- All together, the hosts must have the required memory and CPU resources **in addition to and aside from** what they use to operate or provide to non-OpenShift Container Platform operations.
- The release cycles of OpenShift Container Platform and RHV are different and versions tested might vary in the future depending on the release dates of both products.
- The bootstrap machine provides a temporary control plane while the installation program creates the OpenShift Container Platform cluster. After it creates the cluster, the installation program removes the bootstrap machine and releases its resources.

9.1.4. Verifying the requirements for the RHV environment

Verify that the RHV environment meets the requirements to install and run an OpenShift Container Platform cluster. Not meeting these requirements can cause failures.



IMPORTANT

These requirements are based on the default resources the installation program uses to create control plane and compute machines. These resources include vCPUs, memory, and storage. If you change these resources or increase the number of OpenShift Container Platform machines, adjust these requirements accordingly.

Procedure

1. Check the RHV version.
 - a. In the RHV Administration Portal, click the ? help icon in the upper-right corner and select **About**.

- b. In the window that opens, confirm that the **RHV Software Version** is **4.3.10** or higher.
 2. Inspect the data center, cluster, and storage.
 - a. In the RHV Administration Portal, click **Compute → Data Centers**.
 - b. Confirm the data center where you plan to install OpenShift Container Platform displays a green up arrow, meaning it is "Up".
 - c. Click the name of that data center.
 - d. In the data center details, on the **Storage** tab, confirm the storage domain where you plan to install OpenShift Container Platform is **Active**.
 - e. Record the **Domain Name** for use later on.
 - f. Confirm **Free Space** has at least 230 GiB.
 - g. Confirm that the storage domain meets [these etcd backend performance requirements](#), which can be [measured using the fio performance benchmarking tool](#).
 - h. In the data center details, click the **Clusters** tab.
 - i. Find the RHV cluster where you plan to install OpenShift Container Platform. Record the cluster name for use later on.
 3. Inspect the RHV host resources.
 - a. In the RHV Administration Portal, click **Compute > Clusters**.
 - b. Click the cluster where you plan to install OpenShift Container Platform.
 - c. In the cluster details, click the **Hosts** tab.
 - d. Inspect the hosts and confirm they have a combined total of at least 28 **Logical CPU Cores** available *exclusively* for the OpenShift Container Platform cluster.
 - e. Record the number of available **Logical CPU Cores** for use later on.
 - f. Confirm that these CPU cores are distributed so each of the seven virtual machines created during installation can have four cores.
 - g. Confirm that, all together, the hosts have 112 GiB of **Max free Memory for scheduling new VMs** distributed to meet the requirements for each of the following OpenShift Container Platform machines:
 - 16 GiB required for the bootstrap machine
 - 16 GiB required for each of the three control plane machines
 - 16 GiB for each of the three compute machines
 - h. Record the amount of **Max free Memory for scheduling new VMs** for use later on.
 4. Verify that the virtual network for installing OpenShift Container Platform has access to the RHV Manager's REST API. From a virtual machine on this network, use a curl command with the RHV Manager's REST API. Use the following format:

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

- 1 For **<username>**, specify the user name of an RHV administrator. For **<profile>**, specify the login profile, which you can get by going to the RHV Administration Portal login page and reviewing the **Profile** dropdown list. For **<password>**, specify the admin password.
- 2 For **<engine-fqdn>**, specify the fully qualified domain name of the RHV environment.

For example:

```
$ curl -k -u rhvadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

9.1.5. Preparing the network environment on RHV

Configure three static IP addresses for the OpenShift Container Platform cluster and create DNS entries using two of these addresses.

Procedure

1. Reserve three static IP addresses
 - a. On the network where you plan to install OpenShift Container Platform, identify three static IP addresses that are outside the DHCP lease pool.
 - b. Connect to a host on this network and verify that each of the IP addresses is not in use. For example, use Address Resolution Protocol (ARP) to check that none of the IP addresses have entries:

```
$ arp 10.35.1.19
```

Example output

```
10.35.1.19 (10.35.1.19) -- no entry
```

- c. Reserve three static IP addresses following the standard practices for your network environment.
 - d. Record these IP addresses for future reference.
2. Create DNS entries for the OpenShift Container Platform REST API and apps domain names using this format:

```
api.<cluster-name>.<base-domain> <ip-address> 1
*.apps.<cluster-name>.<base-domain> <ip-address> 2
```

- 1 For **<cluster-name>**, **<base-domain>**, and **<ip-address>**, specify the cluster name, base domain, and static IP address of your OpenShift Container Platform API.
- 2 Specify the cluster name, base domain, and static IP address of your OpenShift Container Platform apps for Ingress and the load balancer.

For example:

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```



NOTE

The third static IP address does not require a DNS entry. The OpenShift Container Platform cluster uses that address for its internal DNS service.

9.1.6. Setting up the CA certificate for RHV

Download the CA certificate from the Red Hat Virtualization (RHV) Manager and set it up on the installation machine.

You can download the certificate from a webpage on the RHV Manager or by using a **curl** command.

Later, you provide the certificate to the installation program.

Procedure

1. Use either of these two methods to download the CA certificate:

- Go to the Manager's webpage, <https://<engine-fqdn>/ovirt-engine/>. Then, under **Downloads**, click the **CA Certificate** link.
- Run the following command:

```
$ curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem 1
```

- 1** For **<engine-fqdn>**, specify the fully qualified domain name of the RHV Manager, such as **rhv-env.virtlab.example.com**.

2. Configure the CA file to grant rootless user access to the Manager. Set the CA file permissions to have an octal value of **0644** (symbolic value: **-rw-r--r--**):

```
$ sudo chmod 0644 /tmp/ca.pem
```

3. For Linux, copy the CA certificate to the directory for server certificates. Use **-p** to preserve the permissions:

```
$ sudo cp -p /tmp/ca.pem /etc/pki/ca-trust/source/anchors/ca.pem
```

4. Add the certificate to the certificate manager for your operating system:

- For macOS, double-click the certificate file and use the **Keychain Access** utility to add the file to the **System** keychain.
- For Linux, update the CA trust:

```
$ sudo update-ca-trust
```

**NOTE**

If you use your own certificate authority, make sure the system trusts it.

Additional Resources

To learn more, see [Authentication and Security](#) in the RHV documentation.

9.1.7. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.

**NOTE**

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```


If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

9.1.8. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

**IMPORTANT**

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

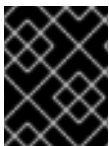
3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

9.1.9. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Open the **ovirt-imageio** port to the Engine from the machine running the installer. By default, the port is **54322**.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

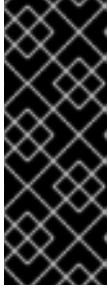
1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCPLOUD_KEYFILE_JSON** environment variables
 - The **~/gcp/osServiceAccount.json** file
 - The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1  
--log-level=info 2
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Respond to the installation program prompts.

- a. Optional: For **SSH Public Key**, select a password-less public key, such as `~/.ssh/id_rsa.pub`. This key authenticates connections with the new OpenShift Container Platform cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, select an SSH key that your **ssh-agent** process uses.

- b. For **Platform**, select **ovirt**.
- c. For **Enter oVirt's API endpoint URL**, enter the URL of the RHV API using this format:

```
https://<engine-fqdn>/ovirt-engine/api 1
```

- 1 For **<engine-fqdn>**, specify the fully qualified domain name of the RHV environment.

For example:

```
$ curl -k -u ovirtadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

- d. For **Is the oVirt CA trusted locally?**, enter **Yes** since you have already set up a CA certificate. Otherwise, enter **No**.
- e. For **oVirt's CA bundle**, if you entered **Yes** for the preceding question, copy the certificate content from `/etc/pki/ca-trust/source/anchors/ca.pem` and paste it here. Then, press **Enter** twice. Otherwise, if you entered **No** for the preceding question, this question does not appear.
- f. For **oVirt engine username**, enter the user name and profile of the RHV administrator using this format:

```
<username>@<profile> 1
```

- 1 For **<username>**, specify the user name of an RHV administrator. For **<profile>**, specify the login profile, which you can get by going to the RHV Administration Portal login page and reviewing the **Profile** dropdown list. Together, the user name and

profile should look similar to this example:

```
admin@internal
```

- g. For **oVirt engine password**, enter the RHV admin password.
- h. For **oVirt cluster**, select the cluster for installing OpenShift Container Platform.
- i. For **oVirt storage domain**, select the storage domain for installing OpenShift Container Platform.
- j. For **oVirt network**, select a virtual network that has access to the RHV Manager REST API.
- k. For **Internal API Virtual IP**, enter the static IP address you set aside for the cluster's REST API.
- l. For **Internal DNS Virtual IP**, enter the static IP address you set aside for the cluster's internal DNS service.
- m. For **Ingress virtual IP**, enter the static IP address you reserved for the wildcard apps domain.
- n. For **Base Domain**, enter the base domain of the OpenShift Container Platform cluster. If this cluster is exposed to the outside world, this must be a valid domain recognized by DNS infrastructure. For example, enter: **virtlab.example.com**
- o. For **Cluster Name**, enter the name of the cluster. For example, **my-cluster**. Use cluster name from the externally registered/resolvable DNS entries you created for the OpenShift Container Platform REST API and apps domain names. The installation program also gives this name to the cluster in the RHV environment.
- p. For **Pull Secret**, copy the pull secret from the **pull-secret.txt** file you downloaded earlier and paste it here. You can also get a copy of the same pull secret from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

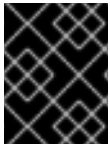


IMPORTANT

You have completed the steps required to install the cluster. The remaining steps show you how to verify the cluster and troubleshoot the installation.

CHAPTER 10. INSTALLING THE CLI BY DOWNLOADING THE BINARY

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

10.1. INSTALLING THE CLI ON LINUX

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

10.2. INSTALLING THE CLI ON WINDOWS

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

10.3. INSTALLING THE CLI ON MACOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

To learn more, see [Getting started with the CLI](#).

CHAPTER 11. LOGGING IN TO THE CLUSTER

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

11.1. VERIFYING CLUSTER STATUS

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:


```
$ oc get clusterversion
```

4. View your Operators' status:

```
$ oc get clusteroperator
```

5. View all running pods in the cluster:

```
$ oc get pods -A
```

Troubleshooting

If the installation fails, the installation program times out and displays an error message. To learn more, see [Troubleshooting installation issues](#).

11.2. ACCESSING THE OPENSIFT CONTAINER PLATFORM WEB CONSOLE ON RHV

After the OpenShift Container Platform cluster initializes, you can log into the OpenShift Container Platform web console.

Procedure

1. Optional: In the Red Hat Virtualization (RHV) Administration Portal, open **Compute → Cluster**.
2. Verify that the installation program creates the virtual machines.
3. Return to the command line where the installation program is running. When the installation program finishes, it displays the user name and temporary password for logging into the OpenShift Container Platform web console.
4. In a browser, open the URL of the OpenShift Container Platform web console. The URL uses this format:

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

- 1** For **<clustername>.<basedomain>**, specify the cluster name and base domain.

For example:

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

11.3. TROUBLESHOOTING COMMON ISSUES WITH INSTALLING ON RED HAT VIRTUALIZATION (RHV)

Here are some common issues you might encounter, along with proposed causes and solutions.

11.3.1. CPU load increases and nodes go into a Not Ready state

- **Symptom:** CPU load increases significantly and nodes start going into a **Not Ready** state.

- **Cause:** The storage domain latency might be too high, especially for master nodes.
- **Solution:**
Make the nodes ready again by restarting the kubelet service. Enter:

```
$ systemctl restart kubelet
```

Inspect the OpenShift Container Platform metrics service, which automatically gathers and reports on some valuable data such as the etcd disk sync duration. If the cluster is operational, use this data to help determine whether storage latency or throughput is the root issue. If so, consider using a storage resource that has lower latency and higher throughput.

To get raw metrics, enter the following command as kubeadmin or user with cluster-admin privileges:

```
$ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics`
```

To learn more, see [Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x](#)

11.3.2. Trouble connecting the OpenShift Container Platform cluster API

- **Symptom:** The installation program completes but the OpenShift Container Platform cluster API is not available. The bootstrap virtual machine remains up after the bootstrap process is complete. When you enter the following command, the response will time out.

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- **Cause:** The bootstrap VM was not deleted by the installation program and has not released the cluster's API IP address.
- **Solution:** Use the **wait-for** subcommand to be notified when the bootstrap process is complete:

```
$ ./openshift-install wait-for bootstrap-complete
```

When the bootstrap process is complete, delete the bootstrap virtual machine:

```
$ ./openshift-install destroy bootstrap
```

11.4. POST-INSTALLATION TASKS

After the OpenShift Container Platform cluster initializes, you can perform the following tasks.

- Optional: After deployment, add or replace SSH keys using the Machine Config Operator (MCO) in OpenShift Container Platform.
- Optional: Remove the **kubeadmin** user. Instead, use the authentication provider to create a user with cluster-admin privileges.

11.5. INSTALLING A CLUSTER ON RHV WITH CUSTOMIZATIONS

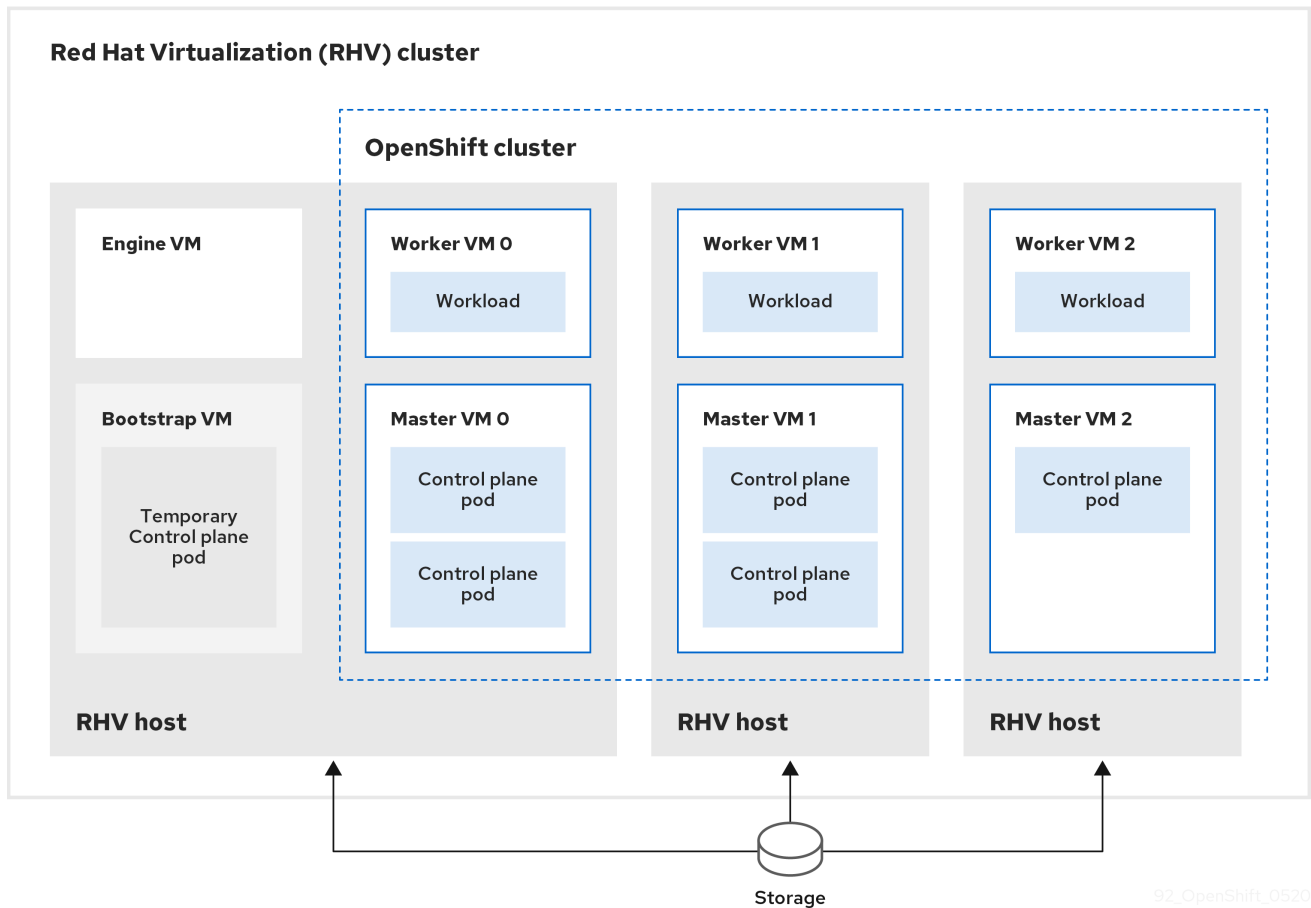
**WARNING**

Due to a known issue with installing OpenShift Container Platform versions 4.4 and 4.5 on Red Hat Virtualization (RHV) 4.4.1, you must customize **install-config.yaml** as described in [OpenShift IPI installation on RHV-4.x failed with "Error: timeout while waiting for state to become 'up' \(last state: 'down', timeout: 10m0s\)"](#). This defect is fixed in RHV 4.4.2.

**WARNING**

Installing OpenShift Container Platform version 4.6 on Red Hat Virtualization (RHV) requires RHV version 4.4. If you are running an earlier version of OpenShift Container Platform on RHV 4.3, do not update it to OpenShift Container Platform version 4.6. Red Hat has not tested running OpenShift Container Platform version 4.6 on RHV version 4.3 and does not support this combination. For more information, see [OpenShift Container Platform 4.x Tested Integrations \(for x86_x64\)](#).

You can customize and install an OpenShift Container Platform cluster on Red Hat Virtualization (RHV), similar to the one shown in the following diagram.



92_OpenShift_0520

The installation program uses installer-provisioned infrastructure to automate creating and deploying the cluster.

To install a customized cluster, you prepare the environment and perform the following steps:

1. Create an installation configuration file, the **install-config.yaml** file, by running the installation program and answering its prompts.
2. Inspect and modify parameters in the **install-config.yaml** file.
3. Make a working copy of the **install-config.yaml** file.
4. Run the installation program with a copy of the **install-config.yaml** file.

Then, the installation program creates the OpenShift Container Platform cluster.

For an alternative to installing a customized cluster, see [Installing a default cluster](#).



NOTE

This installation program is available for Linux and macOS only.

11.5.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, [configure it to allow the sites](#) that your cluster requires access to.

11.5.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

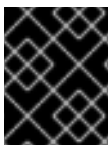
If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

11.5.3. Requirements for the RHV environment

To install and run an OpenShift Container Platform cluster, the RHV environment must meet the following requirements. Not meeting these requirements can cause failures.

The following requirements for CPUs, memory, and storage are based on *default* values multiplied by the default number of virtual machines the installation program creates.

By default, the installation program creates seven machines during the installation process, which includes one bootstrap machine. When the installation program finishes, it deletes the bootstrap machine and frees up its resources. If you perform a custom installation, you can increase the number of virtual machines the installation program creates.



IMPORTANT

If you increase the number of virtual machines or resources in the `install_config.yaml` file, you must also increase these requirements.

Requirements

- The RHV version is 4.3.10 or later.
- The RHV environment has one data center whose state is **Up**.
- The RHV data center contains an RHV cluster.

- The RHV cluster has the following resources exclusively for the OpenShift Container Platform cluster:
 - Minimum 28 vCPUs, which is 4 vCPUs for each of the seven virtual machines created during installation.
 - 112 GiB RAM or more, including:
 - 16 GiB or more for the bootstrap machine, which provides the temporary control plane.
 - 16 GiB or more for each of the three control plane machines which provide the control plane.
 - 16 GiB or more for each of the three compute machines, which run the application workloads.
- The RHV storage domain must meet [these etcd backend performance requirements](#).
- In production environments, each virtual machine must have 120 GiB or more, so the storage domain must have 840 GiB or more for the OpenShift Container Platform cluster. In resource-constrained or non-production environments, each virtual machine must have 32 GiB or more, so the storage domain must have 230 GiB or more for the OpenShift Container Platform cluster.
- The RHV cluster must have access to an Internet connection to download images from the Red Hat Ecosystem Catalog during installation and updates, and for the Telemetry service to simplify the subscription and entitlement process.
- The RHV cluster has a virtual network with access to the REST API on the RHV Manager. Ensure that DHCP is enabled on this network, because the VMs that the installer creates obtain their IP address by using DHCP.

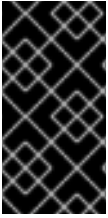


NOTE

- All together, the hosts must have the required memory and CPU resources **in addition to and aside from** what they use to operate or provide to non-OpenShift Container Platform operations.
- The release cycles of OpenShift Container Platform and RHV are different and versions tested might vary in the future depending on the release dates of both products.
- The bootstrap machine provides a temporary control plane while the installation program creates the OpenShift Container Platform cluster. After it creates the cluster, the installation program removes the bootstrap machine and releases its resources.

11.5.4. Verifying the requirements for the RHV environment

Verify that the RHV environment meets the requirements to install and run an OpenShift Container Platform cluster. Not meeting these requirements can cause failures.



IMPORTANT

These requirements are based on the default resources the installation program uses to create control plane and compute machines. These resources include vCPUs, memory, and storage. If you change these resources or increase the number of OpenShift Container Platform machines, adjust these requirements accordingly.

Procedure

1. Check the RHV version.
 - a. In the RHV Administration Portal, click the ? help icon in the upper-right corner and select **About**.
 - b. In the window that opens, confirm that the **RHV Software Version** is **4.3.10** or higher.
2. Inspect the data center, cluster, and storage.
 - a. In the RHV Administration Portal, click **Compute → Data Centers**.
 - b. Confirm the data center where you plan to install OpenShift Container Platform displays a green up arrow, meaning it is "Up".
 - c. Click the name of that data center.
 - d. In the data center details, on the **Storage** tab, confirm the storage domain where you plan to install OpenShift Container Platform is **Active**.
 - e. Record the **Domain Name** for use later on.
 - f. Confirm **Free Space** has at least 230 GiB.
 - g. Confirm that the storage domain meets [these etcd backend performance requirements](#), which can be [measured using the fio performance benchmarking tool](#).
 - h. In the data center details, click the **Clusters** tab.
 - i. Find the RHV cluster where you plan to install OpenShift Container Platform. Record the cluster name for use later on.
3. Inspect the RHV host resources.
 - a. In the RHV Administration Portal, click **Compute > Clusters**.
 - b. Click the cluster where you plan to install OpenShift Container Platform.
 - c. In the cluster details, click the **Hosts** tab.
 - d. Inspect the hosts and confirm they have a combined total of at least 28 **Logical CPU Cores** available *exclusively* for the OpenShift Container Platform cluster.
 - e. Record the number of available **Logical CPU Cores** for use later on.
 - f. Confirm that these CPU cores are distributed so each of the seven virtual machines created during installation can have four cores.

- g. Confirm that, all together, the hosts have 112 GiB of **Max free Memory for scheduling new VMs** distributed to meet the requirements for each of the following OpenShift Container Platform machines:
 - 16 GiB required for the bootstrap machine
 - 16 GiB required for each of the three control plane machines
 - 16 GiB for each of the three compute machines
 - h. Record the amount of **Max free Memory for scheduling new VMs** for use later on.
4. Verify that the virtual network for installing OpenShift Container Platform has access to the RHV Manager's REST API. From a virtual machine on this network, use a curl command with the RHV Manager's REST API. Use the following format:

```
$ curl -k -u <username>@<profile>:<password> \ 1  
https://<engine-fqdn>/ovirt-engine/api 2
```

- 1 For **<username>**, specify the user name of an RHV administrator. For **<profile>**, specify the login profile, which you can get by going to the RHV Administration Portal login page and reviewing the **Profile** dropdown list. For **<password>**, specify the admin password.
- 2 For **<engine-fqdn>**, specify the fully qualified domain name of the RHV environment.

For example:

```
$ curl -k -u rhvadmin@internal:pw123 \  
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

11.5.5. Preparing the network environment on RHV

Configure three static IP addresses for the OpenShift Container Platform cluster and create DNS entries using two of these addresses.

Procedure

1. Reserve three static IP addresses
 - a. On the network where you plan to install OpenShift Container Platform, identify three static IP addresses that are outside the DHCP lease pool.
 - b. Connect to a host on this network and verify that each of the IP addresses is not in use. For example, use Address Resolution Protocol (ARP) to check that none of the IP addresses have entries:

```
$ arp 10.35.1.19
```

Example output

```
10.35.1.19 (10.35.1.19) -- no entry
```


- c. Reserve three static IP addresses following the standard practices for your network environment.
 - d. Record these IP addresses for future reference.
2. Create DNS entries for the OpenShift Container Platform REST API and apps domain names using this format:

```
api.<cluster-name>.<base-domain> <ip-address> 1
*.apps.<cluster-name>.<base-domain> <ip-address> 2
```

- 1 For **<cluster-name>**, **<base-domain>**, and **<ip-address>**, specify the cluster name, base domain, and static IP address of your OpenShift Container Platform API.
- 2 Specify the cluster name, base domain, and static IP address of your OpenShift Container Platform apps for Ingress and the load balancer.

For example:

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```



NOTE

The third static IP address does not require a DNS entry. The OpenShift Container Platform cluster uses that address for its internal DNS service.

11.5.6. Setting up the CA certificate for RHV

Download the CA certificate from the Red Hat Virtualization (RHV) Manager and set it up on the installation machine.

You can download the certificate from a webpage on the RHV Manager or by using a **curl** command.

Later, you provide the certificate to the installation program.

Procedure

1. Use either of these two methods to download the CA certificate:
 - Go to the Manager's webpage, <https://<engine-fqdn>/ovirt-engine/>. Then, under **Downloads**, click the **CA Certificate** link.
 - Run the following command:

```
$ curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem 1
```

- 1 For **<engine-fqdn>**, specify the fully qualified domain name of the RHV Manager, such as **rhv-env.virtlab.example.com**.
2. Configure the CA file to grant rootless user access to the Manager. Set the CA file permissions to have an octal value of **0644** (symbolic value: **-rw-r--r--**):

```
$ sudo chmod 0644 /tmp/ca.pem
```

- For Linux, copy the CA certificate to the directory for server certificates. Use **-p** to preserve the permissions:

```
$ sudo cp -p /tmp/ca.pem /etc/pki/ca-trust/source/anchors/ca.pem
```

- Add the certificate to the certificate manager for your operating system:
 - For macOS, double-click the certificate file and use the **Keychain Access** utility to add the file to the **System** keychain.
 - For Linux, update the CA trust:

```
$ sudo update-ca-trust
```



NOTE

If you use your own certificate authority, make sure the system trusts it.

Additional Resources

To learn more, see [Authentication and Security](#) in the RHV documentation.

11.5.7. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.

Procedure

- If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

11.5.8. Obtaining the installation program

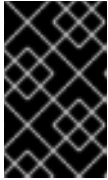
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

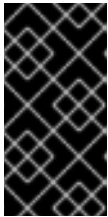
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

11.5.9. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). Red Hat Virtualization (RHV).



WARNING

Due to a known issue with installing OpenShift Container Platform versions 4.4 and 4.5 on Red Hat Virtualization (RHV) 4.4.1, you must customize **install-config.yaml** as described in [OpenShift IPI installation on RHV-4.x failed with "Error: timeout while waiting for state to become 'up' \(last state: 'down', timeout: 10m0s\)"](#). This defect is fixed in RHV 4.4.2.



WARNING

Installing OpenShift Container Platform (OCP) version 4.6 on Red Hat Virtualization (RHV) requires RHV version 4.4. If you are running an earlier version of OCP on RHV 4.3, do not update it to OCP version 4.6. Red Hat has not tested running OCP version 4.6 on RHV version 4.3 and does not support this combination. Also see [OpenShift Container Platform 4.x Tested Integrations \(for x86_x64\)](#).

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.

- a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- i. Select **gcp** as the platform to target.
 - ii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
 - iii. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
 - iv. Select the region to deploy the cluster to.
 - v. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- b. Respond to the installation program prompts.
 - i. For **SSH Public Key**, select a password-less public key, such as `~/.ssh/id_rsa.pub`. This

- i. For **SSH Public Key**, select a password-less public key, such as `~/.ssh/id_rsa.pub`. This key authenticates connections with the new OpenShift Container Platform cluster.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, select an SSH key that your **ssh-agent** process uses.

- ii. For **Platform**, select **ovirt**.
- iii. For **Enter oVirt's API endpoint URL**, enter the URL of the RHV API using this format:

```
https://<engine-fqdn>/ovirt-engine/api 1
```

- 1** For **<engine-fqdn>**, specify the fully qualified domain name of the RHV environment.

For example:

```
$ curl -k -u ovirtadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

- iv. For **Is the oVirt CA trusted locally?**, enter **Yes** since you have already set up a CA certificate. Otherwise, enter **No**.
- v. For **oVirt's CA bundle**, if you entered **Yes** for the preceding question, copy the certificate content from `/etc/pki/ca-trust/source/anchors/ca.pem` and paste it here. Then, press **Enter** twice. Otherwise, if you entered **No** for the preceding question, this question does not appear.
- vi. For **oVirt engine username**, enter the user name and profile of the RHV administrator using this format:

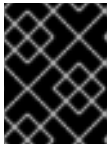
```
<username>@<profile> 1
```

- 1** For **<username>**, specify the user name of an RHV administrator. For **<profile>**, specify the login profile, which you can get by going to the RHV Administration Portal login page and reviewing the **Profile** dropdown list. Together, the user name and profile should look similar to this example:

```
admin@internal
```

- vii. For **oVirt engine password**, enter the RHV admin password.
- viii. For **oVirt cluster**, select the cluster for installing OpenShift Container Platform.
- ix. For **oVirt storage domain**, select the storage domain for installing OpenShift Container Platform.
- x. For **oVirt network**, select a virtual network that has access to the RHV Manager REST API.

- xi. For **Internal API Virtual IP**, enter the static IP address you set aside for the cluster's REST API.
 - xii. For **Internal DNS Virtual IP**, enter the static IP address you set aside for the cluster's internal DNS service.
 - xiii. For **Ingress virtual IP**, enter the static IP address you reserved for the wildcard apps domain.
 - xiv. For **Base Domain**, enter the base domain of the OpenShift Container Platform cluster. If this cluster is exposed to the outside world, this must be a valid domain recognized by DNS infrastructure. For example, enter: **virtlab.example.com**
 - xv. For **Cluster Name**, enter the name of the cluster. For example, **my-cluster**. Use cluster name from the externally registered/resolvable DNS entries you created for the OpenShift Container Platform REST API and apps domain names. The installation program also gives this name to the cluster in the RHV environment.
 - xvi. For **Pull Secret**, copy the pull secret from the **pull-secret.txt** file you downloaded earlier and paste it here. You can also get a copy of the same pull secret from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

11.5.9.1. Example **install-config.yaml** files for Red Hat Virtualization (RHV)

You can customize the OpenShift Container Platform cluster the installation program creates by changing the parameters and parameter values in the **install-config.yaml** file.

The following example is specific to installing OpenShift Container Platform on RHV.

This file is located in the **<installation_directory>** you specified when you ran the following command.

```
$ ./openshift-install create install-config --dir=<installation_directory>
```



NOTE

- These example files are provided for reference only. You must obtain your **install-config.yaml** file by using the installation program.
- Changing the **install-config.yaml** file can increase the resources your cluster requires. Verify that your RHV environment has those additional resources. Otherwise, the installation or cluster will fail.

Example: This is the default **install-config.yaml file**

```
apiVersion: v1
baseDomain: example.com
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: my-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 192.168.1.5
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
  publish: External
  pullSecret: '{"auths": ...}'
  sshKey: ssh-ed12345 AAAA...
```

Example: A minimal install-config.yaml file

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
  pullSecret: '{"auths": ...}'
  sshKey: ssh-ed12345 AAAA...
```


Example: Custom machine pools in an `install-config.yaml` file

```

apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 2
      memoryMB: 65536
      osDisk:
        sizeGB: 100
      vmType: high_performance
  replicas: 3
compute:
- name: worker
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 4
      memoryMB: 65536
      osDisk:
        sizeGB: 200
      vmType: high_performance
  replicas: 5
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...

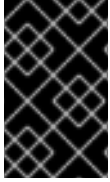
```

11.5.9.2. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **`install-config.yaml`** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **`install-config.yaml`** file to provide more details about the platform.

**NOTE**

After installation, you cannot modify these parameters in the **`install-config.yaml`** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

11.5.9.2.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 11.1. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .


Parameter	Description	Values
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

11.5.9.2.2. Network configuration parameters


You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 11.2. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	<p>Object</p>  <p>NOTE</p> <p>You cannot modify parameters specified by the networking object after installation.</p>

Parameter	Description	Values
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork	The IP address block for services. The default value is 172.30.0.0/16 . The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.	An array with an IP address block in CIDR format. For example: <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>


Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 .  NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.


11.5.9.2.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 11.3. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
compute.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String

Parameter	Description	Values
controlPlane.hyperthreading	<p>Whether to enable or disable simultaneous multithreading, or hyperthreading, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</p>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p>	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere, or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: center;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

11.5.9.2.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 11.4. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.

Parameter	Description	Values
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

11.5.9.2.5. Additional Red Hat Virtualization (RHV) configuration parameters

Additional RHV configuration parameters are described in the following table:

Table 11.5. Additional RHV parameters for clusters

Parameter	Description	Values
platform.ovirt.ovirt_cluster_id	Required. The Cluster where the VMs will be created.	String. For example: 68833f9f-e89c-4891-b768-e2ba0815b76b
platform.ovirt.ovirt_storage_domain_id	Required. The Storage Domain ID where the VM disks will be created.	String. For example: ed7b0f4e-0e96-492a-8fff-279213ee1468
platform.ovirt.ovirt_network_name	Required. The network name where the VM nics will be created.	String. For example: ocpcluster
platform.ovirt.vnicProfileID	Required. The vNIC profile ID of the VM network interfaces. This can be inferred if the cluster network has a single profile.	String. For example: 3fa86930-0be5-4052-b667-b79f0a729692
platform.ovirt.api_vip	Required. An IP address on the machine network that will be assigned to the API virtual IP (VIP). You can access the OpenShift API at this endpoint.	String. Example: 10.46.8.230
platform.ovirt.ingress_vip	Required. An IP address on the machine network that will be assigned to the Ingress virtual IP (VIP).	String. Example: 10.46.8.232

11.5.9.2.6. Additional RHV parameters for machine pools

Additional RHV configuration parameters for machine pools are described in the following table:

Table 11.6. Additional RHV parameters for machine pools

Parameter	Description	Values
<code><machine-pool>.platform.ovirt.cpu</code>	Optional. Defines the CPU of the VM.	Object
<code><machine-pool>.platform.ovirt.cpu.cores</code>	Required if you use <code><machine-pool>.platform.ovirt.cpu</code> . The number of cores. Total virtual CPUs (vCPUs) is cores * sockets.	Integer
<code><machine-pool>.platform.ovirt.cpu.sockets</code>	Required if you use <code><machine-pool>.platform.ovirt.cpu</code> . The number of sockets per core. Total virtual CPUs (vCPUs) is cores * sockets.	Integer
<code><machine-pool>.platform.ovirt.memoryMB</code>	Optional. Memory of the VM in MiB.	Integer
<code><machine-pool>.platform.ovirt.instanceTypeID</code>	Optional. An instance type UUID, such as <code>00000009-0009-0009-0009-0000000000f1</code> , which you can get from the <a href="https://<engine-fqdn>/ovirt-engine/api/instancetype">https://<engine-fqdn>/ovirt-engine/api/instancetype endpoint.	String of UUID
<code><machine-pool>.platform.ovirt.osDisk</code>	Optional. Defines the first and bootable disk of the VM.	String
<code><machine-pool>.platform.ovirt.osDisk.sizeGB</code>	Required if you use <code><machine-pool>.platform.ovirt.osDisk</code> . Size of the disk in GiB.	Number
<code><machine-pool>.platform.ovirt.vmType</code>	Optional. The VM workload type, such as <code>high-performance</code> , <code>server</code> , or <code>desktop</code> .	String



NOTE

You can replace `<machine-pool>` with `controlPlane` or `compute`.

11.5.10. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Open the **ovirt-imageio** port to the Engine from the machine running the installer. By default, the port is **54322**.
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCPLOUD_KEYFILE_JSON** environment variables
 - The `~/.gcp/osServiceAccount.json` file
 - The **gcloud cli** default credentials
2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

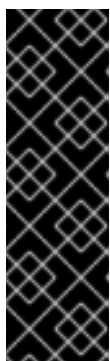
- 1 For **<installation_directory>**, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

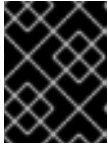
If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



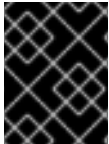
IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrap** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

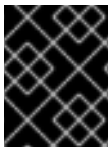
3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

**IMPORTANT**

You have completed the steps required to install the cluster. The remaining steps show you how to verify the cluster and troubleshoot the installation.

11.5.11. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

11.5.11.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

11.5.11.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

11.5.11.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

11.5.12. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container

Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

To learn more, see [Getting started with the CLI](#).

11.5.13. Verifying cluster status

You can verify your OpenShift Container Platform cluster's status during or after installation.

Procedure

1. In the cluster environment, export the administrator's kubeconfig file:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server.

2. View the control plane and compute machines created after a deployment:

```
$ oc get nodes
```

3. View your cluster's version:

```
$ oc get clusterversion
```

4. View your Operators' status:

```
$ oc get clusteroperator
```

5. View all running pods in the cluster:

```
$ oc get pods -A
```

Troubleshooting

If the installation fails, the installation program times out and displays an error message. To learn more, see [Troubleshooting installation issues](#).

11.5.14. Accessing the OpenShift Container Platform web console on RHV

After the OpenShift Container Platform cluster initializes, you can log into the OpenShift Container Platform web console.

Procedure

1. Optional: In the Red Hat Virtualization (RHV) Administration Portal, open **Compute → Cluster**.
2. Verify that the installation program creates the virtual machines.
3. Return to the command line where the installation program is running. When the installation program finishes, it displays the user name and temporary password for logging into the OpenShift Container Platform web console.
4. In a browser, open the URL of the OpenShift Container Platform web console. The URL uses this format:

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

- 1** For **<clustername>.<basedomain>**, specify the cluster name and base domain.

For example:

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

11.5.15. Troubleshooting common issues with installing on Red Hat Virtualization (RHV)

Here are some common issues you might encounter, along with proposed causes and solutions.

11.5.15.1. CPU load increases and nodes go into a **Not Ready** state

- **Symptom:** CPU load increases significantly and nodes start going into a **Not Ready** state.
- **Cause:** The storage domain latency might be too high, especially for master nodes.
- **Solution:**
Make the nodes ready again by restarting the kubelet service. Enter:

■

```
$ systemctl restart kubelet
```

Inspect the OpenShift Container Platform metrics service, which automatically gathers and reports on some valuable data such as the etcd disk sync duration. If the cluster is operational, use this data to help determine whether storage latency or throughput is the root issue. If so, consider using a storage resource that has lower latency and higher throughput.

To get raw metrics, enter the following command as kubeadmin or user with cluster-admin privileges:

```
$ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics`
```

To learn more, see [Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x](#)

11.5.15.2. Trouble connecting the OpenShift Container Platform cluster API

- **Symptom:** The installation program completes but the OpenShift Container Platform cluster API is not available. The bootstrap virtual machine remains up after the bootstrap process is complete. When you enter the following command, the response will time out.

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- **Cause:** The bootstrap VM was not deleted by the installation program and has not released the cluster's API IP address.
- **Solution:** Use the **wait-for** subcommand to be notified when the bootstrap process is complete:

```
$ ./openshift-install wait-for bootstrap-complete
```

When the bootstrap process is complete, delete the bootstrap virtual machine:

```
$ ./openshift-install destroy bootstrap
```

11.5.16. Post-installation tasks

After the OpenShift Container Platform cluster initializes, you can perform the following tasks.

- Optional: After deployment, add or replace SSH keys using the Machine Config Operator (MCO) in OpenShift Container Platform.
- Optional: Remove the **kubeadmin** user. Instead, use the authentication provider to create a user with cluster-admin privileges.

11.5.17. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).

11.6. UNINSTALLING A CLUSTER ON RHV

You can remove an OpenShift Container Platform cluster from Red Hat Virtualization (RHV).

11.6.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access. For example, some Google Cloud resources require [IAM permissions](#) in shared VPC host projects, or there might be unused [health checks that must be deleted](#).

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.
- Have the files that the installation program generated when you created your cluster.

Procedure

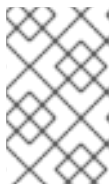
1. From the computer that you used to install the cluster, run the following command:

```

$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2

```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

CHAPTER 12. INSTALLING ON VSPHERE

12.1. INSTALLING A CLUSTER ON VSPHERE

In OpenShift Container Platform version 4.5, you can install a cluster on your VMware vSphere instance by using installer-provisioned infrastructure.

12.1.1. Prerequisites

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

12.1.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

12.1.3. VMware vSphere infrastructure requirements

You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.1. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.1.4. vCenter requirements

Before you install an OpenShift Container Platform cluster on your vCenter that uses infrastructure that the installer provisions, you must prepare your environment.

Required vCenter account privileges

To install an OpenShift Container Platform cluster in a vCenter, the installation program requires access to an account with privileges to read and create the required resources. Using an account that has global administrative privileges is the simplest way to access all of the necessary permissions.

If you cannot use an account with global administrative privileges, you must create roles to grant the privileges necessary for OpenShift Container Platform cluster installation. While most of the privileges are always required, some are required only if you plan for the installation program to provision a folder to contain the OpenShift Container Platform cluster on your vCenter instance, which is the default behavior. You must create or amend vSphere roles for the specified objects to grant the required privileges.

An additional role is required if the installation program is to create a vSphere virtual machine folder.

Example 12.1. Roles and privileges required for installation

vSphere object for role	When required	Required privileges
vSphere vCenter	Always	Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View
vSphere vCenter Cluster	Always	Host.Config.StorageResource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk
vSphere Datastore	Always	Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement
vSphere Port Group	Always	Network.Assign

vSphere object for role	When required	Required privileges
Virtual Machine Folder	Always	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone

vSphere object for role	When required	Required privileges
vSphere vCenter Datacenter	If the installation program creates the virtual machine folder	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone Folder.Create Folder.Delete

Additionally, the user requires some **ReadOnly** permissions, and some of the roles require permission to propagate the permissions to child objects. These settings vary depending on whether or not you install the cluster into an existing folder.

Example 12.2. Required permissions and propagation settings

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter	Always	False	Listed required privileges
vSphere vCenter Datacenter	Existing folder	False	ReadOnly permission
	Installation program creates the folder	True	Listed required privileges
vSphere vCenter Cluster	Always	True	Listed required privileges
vSphere vCenter Datastore	Always	False	Listed required privileges
vSphere Switch	Always	False	ReadOnly permission
vSphere Port Group	Always	False	Listed required privileges
vSphere vCenter Virtual Machine Folder	Existing folder	True	Listed required privileges

For more information about creating an account with only the required privileges, see [vSphere Permissions and User Management Tasks](#) in the vSphere documentation.

Using OpenShift Container Platform with vMotion



IMPORTANT

OpenShift Container Platform generally supports compute-only vMotion. Using Storage vMotion can cause issues and is not supported.

If you are using vSphere volumes in your pods, migrating a VM across datastores either manually or through Storage vMotion causes invalid references within OpenShift Container Platform persistent volume (PV) objects. These references prevent affected pods from starting up and can result in data loss.

Similarly, OpenShift Container Platform does not support selective migration of VMDKs across datastores, using datastore clusters for VM provisioning or for dynamic or static provisioning of PVs, or using a datastore that is part of a datastore cluster for dynamic or static provisioning of PVs.

Cluster resources

When you deploy an OpenShift Container Platform cluster that uses installer-provisioned infrastructure, the installation program must be able to create several resources in your vCenter instance.

A standard OpenShift Container Platform installation creates the following vCenter resources:

- 1 Folder
- 1 Tag category
- 1 Tag
- Virtual machines:
 - 1 template
 - 1 temporary bootstrap node
 - 3 control plane nodes
 - 3 compute machines

Although these resources use 856 GB of storage, the bootstrap node is destroyed during the cluster installation process. A minimum of 800 GB of storage is required to use a standard cluster.

If you deploy more compute machines, the OpenShift Container Platform cluster will use more storage.

Cluster limits

Available resources vary between clusters. The number of possible clusters within a vCenter is limited primarily by available storage space and any limitations on the number of required resources. Be sure to consider both limitations to the vCenter resources that the cluster creates and the resources that you require to deploy a cluster, such as IP addresses and networks.

Networking requirements

You must use DHCP for the network and ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines. Additionally, you must create the following networking resources before you install the OpenShift Container Platform cluster:

Required IP addresses

An installer-provisioned vSphere installation requires two static IP addresses:

- The **API** address is used to access the cluster API.
- The **Ingress** address is used for cluster ingress traffic.

You must provide these IP addresses to the installation program when you install the OpenShift Container Platform cluster.

DNS records

You must create DNS records for two static IP addresses in the appropriate DNS server for the vCenter instance that hosts your OpenShift Container Platform cluster. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify when you install the cluster. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 12.2. Required DNS records

Component	Record	Description
API VIP	api.<cluster_name>.<base_domain>.	This DNS A/AAAA or CNAME record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	A wildcard DNS A/AAAA or CNAME record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

12.1.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

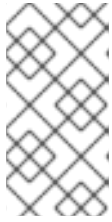
1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

1

Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

12.1.6. Obtaining the installation program

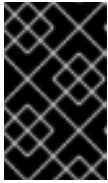
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

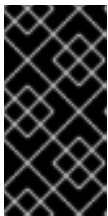
Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

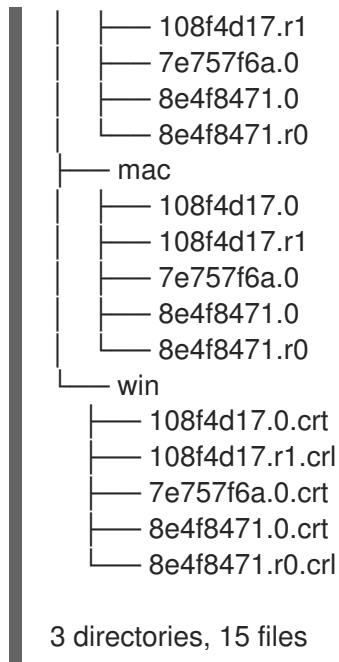
12.1.7. Adding vCenter root CA certificates to your system trust

Because the installation program requires access to your vCenter's API, you must add your vCenter's trusted root CA certificates to your system trust before you install an OpenShift Container Platform cluster.

Procedure

1. From the vCenter home page, download the vCenter's root CA certificates. Click **Download trusted root CA certificates** in the vSphere Web Services SDK section. The **<vCenter>/certs/download.zip** file downloads.
2. Extract the compressed file that contains the vCenter root CA certificates. The contents of the compressed file resemble the following file structure:

```
certs
├── lin
│   └── 108f4d17.0
```



3. Add the files for your operating system to the system trust. For example, on a Fedora operating system, run the following command:

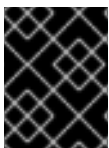
```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. Update your system trust. For example, on a Fedora operating system, run the following command:

```
# update-ca-trust extract
```

12.1.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

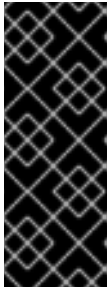
1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCPLOUD_KEYFILE_JSON** environment variables
 - The `~/gcp/osServiceAccount.json` file
 - The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

Provide values at the prompts:

a. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

b. Select **gcp** as the platform to target.

c. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.

d. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.

e. Select the region to deploy the cluster to.

f. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.

g. Select **vsphere** as the platform to target.

h. Specify the name of your vCenter instance.

i. Specify the user name and password for the vCenter account that has the required permissions to create the cluster.
The installation program connects to your vCenter instance.

j. Select the datacenter in your vCenter instance to connect to.

k. Select the default vCenter datastore to use.

- l. Select the vCenter cluster to install the OpenShift Container Platform cluster in.
- m. Select the network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.
- n. Enter the virtual IP address that you configured for control plane API access.
- o. Enter the virtual IP address that you configured for cluster ingress.
- p. Enter the base domain. This base domain must be the same one that you used in the DNS records that you configured.
- q. Enter a descriptive name for your cluster. The cluster name must be the same one that you used in the DNS records that you configured. If you provide a name that is longer than 6 characters, only the first 6 characters will be used in the infrastructure ID that is generated from the cluster name.
- r. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

12.1.9. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

12.1.9.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.1.9.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

12.1.9.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.1.10. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:


```
$ oc whoami
```

Example output

```
system:admin
```

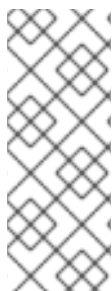
12.1.11. Creating registry storage

After you install the cluster, you must create storage for the registry Operator.

12.1.11.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

12.1.11.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

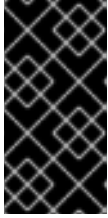
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

12.1.11.2.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

**IMPORTANT**

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.

**IMPORTANT**

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

**NOTE**

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

**NOTE**

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

12.1.11.2.2. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
 - a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
  - ReadWriteOnce 2
  resources:
    requests:
      storage: 100Gi 3
```

- 1 A unique name that represents the **PersistentVolumeClaim** object.
- 2 The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.
- 3 The size of the PersistentVolumeClaim.

- b. Create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:  
  pvc:  
    claim: 1
```

- 1** Creating a custom PVC allows you to leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see [Configuring the registry for vSphere](#).

12.1.12. Backing up VMware vSphere volumes

OpenShift Container Platform provisions new volumes as independent persistent disks to freely attach and detach the volume on any node in the cluster. As a consequence, it is not possible to back up volumes that use snapshots, or to restore volumes from snapshots. See [Snapshot Limitations](#) for more information.

Procedure

To create a backup of persistent volumes:

1. Stop the application that is using the persistent volume.
2. Clone the persistent volume.
3. Restart the application.
4. Create a backup of the cloned volume.
5. Delete the cloned volume.

12.1.13. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- [Set up your registry and configure registry storage](#).

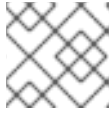
12.2. INSTALLING A CLUSTER ON VSPHERE WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster on your VMware vSphere instance by using installer-provisioned infrastructure. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

12.2.1. Prerequisites

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.

**NOTE**

Be sure to also review this site list if you are configuring a proxy.

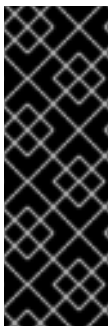
12.2.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

**IMPORTANT**

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

12.2.3. VMware vSphere infrastructure requirements

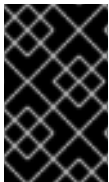
You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.3. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .

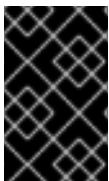
Component	Minimum supported versions	Description
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.2.4. vCenter requirements

Before you install an OpenShift Container Platform cluster on your vCenter that uses infrastructure that the installer provisions, you must prepare your environment.

Required vCenter account privileges

To install an OpenShift Container Platform cluster in a vCenter, the installation program requires access to an account with privileges to read and create the required resources. Using an account that has global administrative privileges is the simplest way to access all of the necessary permissions.

If you cannot use an account with global administrative privileges, you must create roles to grant the privileges necessary for OpenShift Container Platform cluster installation. While most of the privileges are always required, some are required only if you plan for the installation program to provision a folder to contain the OpenShift Container Platform cluster on your vCenter instance, which is the default behavior. You must create or amend vSphere roles for the specified objects to grant the required privileges.

An additional role is required if the installation program is to create a vSphere virtual machine folder.

Example 12.3. Roles and privileges required for installation

vSphere object for role	When required	Required privileges
vSphere vCenter	Always	Cns.Searchable InventoryService.Tagging.A ttachTag InventoryService.Tagging.C reateCategory InventoryService.Tagging.C reateTag InventoryService.Tagging.D eleteCategory InventoryService.Tagging.D eleteTag InventoryService.Tagging.E ditCategory InventoryService.Tagging.E ditTag Sessions.ValidateSession StorageProfile.View
vSphere vCenter Cluster	Always	Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.Add NewDisk
vSphere Datastore	Always	Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement
vSphere Port Group	Always	Network.Assign

vSphere object for role	When required	Required privileges
Virtual Machine Folder	Always	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone

vSphere object for role	When required	Required privileges
vSphere vCenter Datacenter	If the installation program creates the virtual machine folder	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone Folder.Create Folder.Delete

vSphere object for role

When required

Required privileges

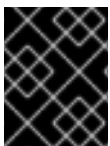
Additionally, the user requires some **ReadOnly** permissions, and some of the roles require permission to propagate the permissions to child objects. These settings vary depending on whether or not you install the cluster into an existing folder.

Example 12.4. Required permissions and propagation settings

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter	Always	False	Listed required privileges
vSphere vCenter Datacenter	Existing folder	False	ReadOnly permission
	Installation program creates the folder	True	Listed required privileges
vSphere vCenter Cluster	Always	True	Listed required privileges
vSphere vCenter Datastore	Always	False	Listed required privileges
vSphere Switch	Always	False	ReadOnly permission
vSphere Port Group	Always	False	Listed required privileges
vSphere vCenter Virtual Machine Folder	Existing folder	True	Listed required privileges

For more information about creating an account with only the required privileges, see [vSphere Permissions and User Management Tasks](#) in the vSphere documentation.

Using OpenShift Container Platform with vMotion



IMPORTANT

OpenShift Container Platform generally supports compute-only vMotion. Using Storage vMotion can cause issues and is not supported.

If you are using vSphere volumes in your pods, migrating a VM across datastores either manually or through Storage vMotion causes invalid references within OpenShift Container Platform persistent volume (PV) objects. These references prevent affected pods from starting up and can result in data loss.

Similarly, OpenShift Container Platform does not support selective migration of VMDKs across datastores, using datastore clusters for VM provisioning or for dynamic or static provisioning of PVs, or using a datastore that is part of a datastore cluster for dynamic or static provisioning of PVs.

Cluster resources

When you deploy an OpenShift Container Platform cluster that uses installer-provisioned infrastructure, the installation program must be able to create several resources in your vCenter instance.

A standard OpenShift Container Platform installation creates the following vCenter resources:

- 1 Folder
- 1 Tag category
- 1 Tag
- Virtual machines:
 - 1 template
 - 1 temporary bootstrap node
 - 3 control plane nodes
 - 3 compute machines

Although these resources use 856 GB of storage, the bootstrap node is destroyed during the cluster installation process. A minimum of 800 GB of storage is required to use a standard cluster.

If you deploy more compute machines, the OpenShift Container Platform cluster will use more storage.

Cluster limits

Available resources vary between clusters. The number of possible clusters within a vCenter is limited primarily by available storage space and any limitations on the number of required resources. Be sure to consider both limitations to the vCenter resources that the cluster creates and the resources that you require to deploy a cluster, such as IP addresses and networks.

Networking requirements

You must use DHCP for the network and ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines. Additionally, you must create the following networking resources before you install the OpenShift Container Platform cluster:

Required IP addresses

An installer-provisioned vSphere installation requires two static IP addresses:

- The **API** address is used to access the cluster API.
- The **Ingress** address is used for cluster ingress traffic.

You must provide these IP addresses to the installation program when you install the OpenShift Container Platform cluster.

DNS records

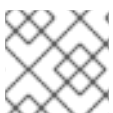
You must create DNS records for two static IP addresses in the appropriate DNS server for the vCenter instance that hosts your OpenShift Container Platform cluster. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify when you install the cluster. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 12.4. Required DNS records

Component	Record	Description
API VIP	api.<cluster_name>.<base_domain>.	This DNS A/AAAA or CNAME record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	A wildcard DNS A/AAAA or CNAME record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

12.2.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

12.2.6. Obtaining the installation program

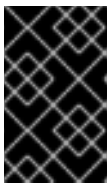
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

12.2.7. Adding vCenter root CA certificates to your system trust

Because the installation program requires access to your vCenter's API, you must add your vCenter's trusted root CA certificates to your system trust before you install an OpenShift Container Platform cluster.

Procedure

1. From the vCenter home page, download the vCenter's root CA certificates. Click **Download trusted root CA certificates** in the vSphere Web Services SDK section. The **<vCenter>/certs/download.zip** file downloads.
2. Extract the compressed file that contains the vCenter root CA certificates. The contents of the compressed file resemble the following file structure:

```

certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

```

3 directories, 15 files

3. Add the files for your operating system to the system trust. For example, on a Fedora operating system, run the following command:

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. Update your system trust. For example, on a Fedora operating system, run the following command:

```
# update-ca-trust extract
```

12.2.8. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). VMware vSphere.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- v. Select the region to deploy the cluster to.
- vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- vii. Select **vsphere** as the platform to target.
- viii. Specify the name of your vCenter instance.
- ix. Specify the user name and password for the vCenter account that has the required permissions to create the cluster.
The installation program connects to your vCenter instance.
- x. Select the datacenter in your vCenter instance to connect to.
- xi. Select the default vCenter datastore to use.
- xii. Select the vCenter cluster to install the OpenShift Container Platform cluster in.
- xiii. Select the network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.
- xiv. Enter the virtual IP address that you configured for control plane API access.
- xv. Enter the virtual IP address that you configured for cluster ingress.

- xvi. Enter the base domain. This base domain must be the same one that you used in the DNS records that you configured.
 - xvii. Enter a descriptive name for your cluster. The cluster name must be the same one that you used in the DNS records that you configured.
 - xviii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

12.2.8.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

12.2.8.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 12.5. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String


Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}.{{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


12.2.8.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 12.6. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .

Parameter	Description	Values
networking.serviceNetwork	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	<p>Required if you use networking.machineNetwork. An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24.</p>	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div>

12.2.8.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 12.7. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

12.2.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 12.8. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

Table 12.9. Additional VMware vSphere cluster parameters

Parameter	Description	Values
platform.vsphere.vCenter	The fully-qualified hostname or IP address of the vCenter server.	String
platform.vsphere.username	The user name to use to connect to the vCenter instance with. This user must have at least the roles and privileges that are required for static or dynamic persistent volume provisioning in vSphere.	String
platform.vsphere.password	The password for the vCenter user name.	String
platform.vsphere.datacenter	The name of the datacenter to use in the vCenter instance.	String
platform.vsphere.defaultDatastore	The name of the default datastore to use for provisioning volumes.	String

Parameter	Description	Values
platform.vsphere.folder	<i>Optional.</i> The absolute path of an existing folder where the installation program creates the virtual machines. If you do not provide this value, the installation program creates a folder that is named with the infrastructure ID in the datacenter virtual machine folder.	String, for example, /<datacenter_name>/vm/<folder_name>/<subfolder_name> .
platform.vsphere.network	The network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.	String
platform.vsphere.cluster	The vCenter cluster to install the OpenShift Container Platform cluster in.	String
platform.vsphere.apiVIP	The virtual IP (VIP) address that you configured for control plane API access.	An IP address, for example 128.0.0.1 .
platform.vsphere.ingressVIP	The virtual IP (VIP) address that you configured for cluster ingress.	An IP address, for example 128.0.0.1 .

12.2.8.1.5. Optional VMware vSphere machine pool configuration parameters

Optional VMware vSphere machine pool configuration parameters are described in the following table:

Table 12.10. Optional VMware vSphere machine pool parameters

Parameter	Description	Values
platform.vsphere.clusterOSImage	The location from which the installer downloads the RHCOS image. You must set this parameter to perform an installation in a restricted network.	An HTTP or HTTPS URL, optionally with a SHA-256 checksum. For example, https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova .
platform.vsphere.osDisk.diskSizeGB	The size of the disk in gigabytes.	Integer
platform.vsphere.cpus	The total number of virtual processor cores to assign a virtual machine.	Integer

Parameter	Description	Values
platform.vsphere.coresPerSocket	The number of cores per socket in a virtual machine. The number of virtual CPUs (vCPUs) on the virtual machine is platform.vsphere.cpus/platform.vsphere.coresPerSocket . The default value is 1	Integer
platform.vsphere.memoryMB	The size of a virtual machine's memory in megabytes.	Integer

12.2.8.2. Sample install-config.yaml file for an installer-provisioned VMware vSphere cluster

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com ❶
compute: ❷
- hyperthreading: Enabled ❸
  name: worker
  replicas: 3
  platform:
    vsphere: ❹
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8196
      osDisk:
        diskSizeGB: 120
controlPlane: ❺
  hyperthreading: Enabled ❻
  name: master
  replicas: 3
  platform:
    vsphere: ❼
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster ❽
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore

```

```

folder: folder
network: VM_Network
cluster: vsphere_cluster_name
apiVIP: api_vip
ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'

```

- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Your machines must use at least 8 CPUs and 32 GB of RAM if you disable simultaneous multithreading.

- 4 7 Optional: Provide additional configuration for the machine pool parameters for the compute and control plane machines.
- 8 The cluster name that you specified in your DNS records.

12.2.9. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:

- The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
- The `~/.gcp/osServiceAccount.json` file
- The **gcloud cli** default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

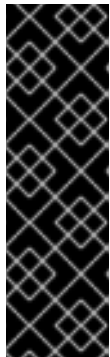
- 1 For `<installation_directory>`, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

12.2.10. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

12.2.10.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.2.10.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

12.2.10.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.2.11. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

12.2.12. Creating registry storage

After you install the cluster, you must create storage for the registry Operator.

12.2.12.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

12.2.12.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

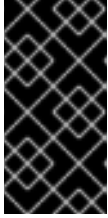
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

12.2.12.2.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

**IMPORTANT**

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.

**IMPORTANT**

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

**NOTE**

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

**NOTE**

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

12.2.12.2.2. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
 - a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
  - ReadWriteOnce 2
  resources:
    requests:
      storage: 100Gi 3
```

- 1 A unique name that represents the **PersistentVolumeClaim** object.
- 2 The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.
- 3 The size of the PersistentVolumeClaim.

- b. Create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:  
  pvc:  
    claim: 1
```

- 1** Creating a custom PVC allows you to leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see [Configuring the registry for vSphere](#).

12.2.13. Backing up VMware vSphere volumes

OpenShift Container Platform provisions new volumes as independent persistent disks to freely attach and detach the volume on any node in the cluster. As a consequence, it is not possible to back up volumes that use snapshots, or to restore volumes from snapshots. See [Snapshot Limitations](#) for more information.

Procedure

To create a backup of persistent volumes:

1. Stop the application that is using the persistent volume.
2. Clone the persistent volume.
3. Restart the application.
4. Create a backup of the cloned volume.
5. Delete the cloned volume.

12.2.14. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- [Set up your registry and configure registry storage](#).

12.3. INSTALLING A CLUSTER ON VSPHERE WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster on your VMware vSphere instance by using installer-provisioned infrastructure with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

12.3.1. Prerequisites

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

12.3.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

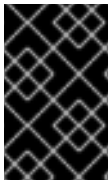
12.3.3. VMware vSphere infrastructure requirements

You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.11. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.3.4. vCenter requirements

Before you install an OpenShift Container Platform cluster on your vCenter that uses infrastructure that the installer provisions, you must prepare your environment.

Required vCenter account privileges

To install an OpenShift Container Platform cluster in a vCenter, the installation program requires access to an account with privileges to read and create the required resources. Using an account that has global administrative privileges is the simplest way to access all of the necessary permissions.

If you cannot use an account with global administrative privileges, you must create roles to grant the privileges necessary for OpenShift Container Platform cluster installation. While most of the privileges

are always required, some are required only if you plan for the installation program to provision a folder to contain the OpenShift Container Platform cluster on your vCenter instance, which is the default behavior. You must create or amend vSphere roles for the specified objects to grant the required privileges.

An additional role is required if the installation program is to create a vSphere virtual machine folder.

Example 12.5. Roles and privileges required for installation

vSphere object for role	When required	Required privileges
vSphere vCenter	Always	Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View
vSphere vCenter Cluster	Always	Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk
vSphere Datastore	Always	Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement
vSphere Port Group	Always	Network.Assign

vSphere object for role	When required	Required privileges
Virtual Machine Folder	Always	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone

vSphere object for role	When required	Required privileges
vSphere vCenter Datacenter	If the installation program creates the virtual machine folder	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone Folder.Create Folder.Delete

Additionally, the user requires some **ReadOnly** permissions, and some of the roles require permission to propagate the permissions to child objects. These settings vary depending on whether or not you install the cluster into an existing folder.

Example 12.6. Required permissions and propagation settings

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter	Always	False	Listed required privileges
vSphere vCenter Datacenter	Existing folder	False	ReadOnly permission
	Installation program creates the folder	True	Listed required privileges
vSphere vCenter Cluster	Always	True	Listed required privileges
vSphere vCenter Datastore	Always	False	Listed required privileges
vSphere Switch	Always	False	ReadOnly permission
vSphere Port Group	Always	False	Listed required privileges
vSphere vCenter Virtual Machine Folder	Existing folder	True	Listed required privileges

For more information about creating an account with only the required privileges, see [vSphere Permissions and User Management Tasks](#) in the vSphere documentation.

Using OpenShift Container Platform with vMotion



IMPORTANT

OpenShift Container Platform generally supports compute-only vMotion. Using Storage vMotion can cause issues and is not supported.

If you are using vSphere volumes in your pods, migrating a VM across datastores either manually or through Storage vMotion causes invalid references within OpenShift Container Platform persistent volume (PV) objects. These references prevent affected pods from starting up and can result in data loss.

Similarly, OpenShift Container Platform does not support selective migration of VMDKs across datastores, using datastore clusters for VM provisioning or for dynamic or static provisioning of PVs, or using a datastore that is part of a datastore cluster for dynamic or static provisioning of PVs.

Cluster resources

When you deploy an OpenShift Container Platform cluster that uses installer-provisioned infrastructure, the installation program must be able to create several resources in your vCenter instance.

A standard OpenShift Container Platform installation creates the following vCenter resources:

- 1 Folder
- 1 Tag category
- 1 Tag
- Virtual machines:
 - 1 template
 - 1 temporary bootstrap node
 - 3 control plane nodes
 - 3 compute machines

Although these resources use 856 GB of storage, the bootstrap node is destroyed during the cluster installation process. A minimum of 800 GB of storage is required to use a standard cluster.

If you deploy more compute machines, the OpenShift Container Platform cluster will use more storage.

Cluster limits

Available resources vary between clusters. The number of possible clusters within a vCenter is limited primarily by available storage space and any limitations on the number of required resources. Be sure to consider both limitations to the vCenter resources that the cluster creates and the resources that you require to deploy a cluster, such as IP addresses and networks.

Networking requirements

You must use DHCP for the network and ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines. Additionally, you must create the following networking resources before you install the OpenShift Container Platform cluster:

Required IP addresses

An installer-provisioned vSphere installation requires two static IP addresses:

- The **API** address is used to access the cluster API.
- The **Ingress** address is used for cluster ingress traffic.

You must provide these IP addresses to the installation program when you install the OpenShift Container Platform cluster.

DNS records

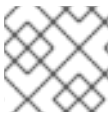
You must create DNS records for two static IP addresses in the appropriate DNS server for the vCenter instance that hosts your OpenShift Container Platform cluster. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify when you install the cluster. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 12.12. Required DNS records

Component	Record	Description
API VIP	api.<cluster_name>.<base_domain>.	This DNS A/AAAA or CNAME record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	A wildcard DNS A/AAAA or CNAME record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

12.3.5. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

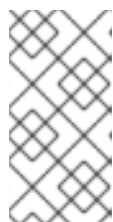
Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

12.3.6. Obtaining the installation program

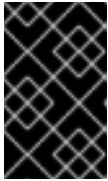
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

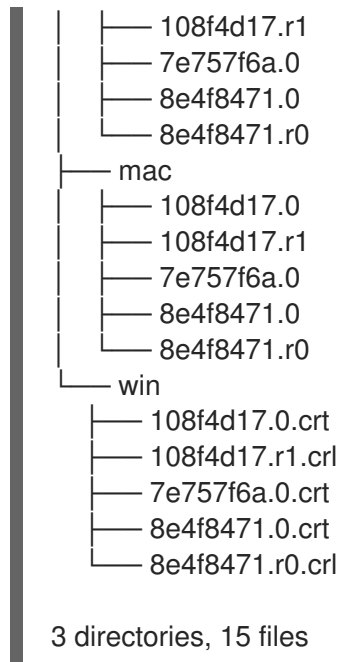
12.3.7. Adding vCenter root CA certificates to your system trust

Because the installation program requires access to your vCenter's API, you must add your vCenter's trusted root CA certificates to your system trust before you install an OpenShift Container Platform cluster.

Procedure

1. From the vCenter home page, download the vCenter's root CA certificates. Click **Download trusted root CA certificates** in the vSphere Web Services SDK section. The **<vCenter>/certs/download.zip** file downloads.
2. Extract the compressed file that contains the vCenter root CA certificates. The contents of the compressed file resemble the following file structure:

```
certs
├── lin
│   └── 108f4d17.0
```



3. Add the files for your operating system to the system trust. For example, on a Fedora operating system, run the following command:

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. Update your system trust. For example, on a Fedora operating system, run the following command:

```
# update-ca-trust extract
```

12.3.8. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). VMware vSphere.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- v. Select the region to deploy the cluster to.
- vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- vii. Select **vsphere** as the platform to target.
- viii. Specify the name of your vCenter instance.
 - ix. Specify the user name and password for the vCenter account that has the required permissions to create the cluster.
The installation program connects to your vCenter instance.
 - x. Select the datacenter in your vCenter instance to connect to.
 - xi. Select the default vCenter datastore to use.
 - xii. Select the vCenter cluster to install the OpenShift Container Platform cluster in.
 - xiii. Select the network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.
 - xiv. Enter the virtual IP address that you configured for control plane API access.
 - xv. Enter the virtual IP address that you configured for cluster ingress.

- xvi. Enter the base domain. This base domain must be the same one that you used in the DNS records that you configured.
 - xvii. Enter a descriptive name for your cluster. The cluster name must be the same one that you used in the DNS records that you configured.
 - xviii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the **Installation configuration parameters** section.
 3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

12.3.8.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

12.3.8.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 12.13. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String


Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}.{{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


12.3.8.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 12.14. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .

Parameter	Description	Values
networking.serviceNetwork	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	<p>Required if you use networking.machineNetwork. An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24.</p>	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div>


12.3.8.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 12.15. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>  <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p>  <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p>	For example, sshKey: ssh-ed25519 AAAA...

12.3.8.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 12.16. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

12.3.8.2. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 12.17. Required network parameters

Parameter	Description	Value
networking.networkType	The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5.	The default value is OpenShiftSDN .
networking.clusterNetwork[].cidr	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .

Parameter	Description	Value
networking.clusterNetwork[].hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork[]	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.machineNetwork[].cidr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

12.3.8.3. Sample install-config.yaml file for an installer-provisioned VMware vSphere cluster

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 3
  platform:
    vsphere: 4
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8196
      osDisk:
        diskSizeGB: 120
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3
  platform:
    vsphere: 7
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster 8

```

```

networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name
    apiVIP: api_vip
    ingressVIP: ingress_vip
  fips: false
  pullSecret: '{"auths": ...}'
  sshKey: 'ssh-ed25519 AAAA...'

```

- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Your machines must use at least 8 CPUs and 32 GB of RAM if you disable simultaneous multithreading.

- 4 7 Optional: Provide additional configuration for the machine pool parameters for the compute and control plane machines.
- 8 The cluster name that you specified in your DNS records.

12.3.9. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster.

Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of [kube-proxy](#) settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.

Procedure

1. Use the following command to create manifests:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1 For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, several network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
```

```

- cidr: 10.128.0.0/14
  hostPrefix: 23
serviceNetwork:
- 172.30.0.0/16
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789

```

- 1 The parameters for the **spec** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.
5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

12.3.10. Cluster Network Operator configuration

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```

apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
  ...
  kubeProxyConfig: 4
  iptablesSyncPeriod: 30s 5
  proxyArguments:
    iptables-min-sync-period: 6
    - 0s

```

- 1 2 Specified in the **install-config.yaml** file.

- 3 Configures the default Container Network Interface (CNI) network provider for the cluster network.
- 4 The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.
- 5 The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.



NOTE

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

- 6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#).

12.3.10.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**.

- 5 The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

12.3.10.2. Cluster Network Operator example configuration

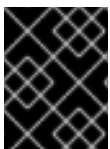
A complete CR object for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 0s
```

12.3.11. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GCPLOUD_KEYFILE_JSON** environment variables

- The `~/gcp/osServiceAccount.json` file
- The `gcloud cli` default credentials

2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

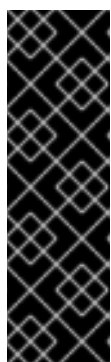
- 1 For `<installation_directory>`, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.



IMPORTANT

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

12.3.12. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

12.3.12.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.3.12.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

12.3.12.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.3.13. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

12.3.14. Creating registry storage

After you install the cluster, you must create storage for the registry Operator.

12.3.14.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

12.3.14.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

12.3.14.2.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

**IMPORTANT**

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.

**IMPORTANT**

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

**NOTE**

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```

**NOTE**

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

12.3.14.2.2. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
 - a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
  - ReadWriteOnce 2
  resources:
    requests:
      storage: 100Gi 3
```

- 1 A unique name that represents the **PersistentVolumeClaim** object.
- 2 The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.
- 3 The size of the PersistentVolumeClaim.

- b. Create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Creating a custom PVC allows you to leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see [Configuring the registry for vSphere](#).

12.3.15. Backing up VMware vSphere volumes

OpenShift Container Platform provisions new volumes as independent persistent disks to freely attach and detach the volume on any node in the cluster. As a consequence, it is not possible to back up volumes that use snapshots, or to restore volumes from snapshots. See [Snapshot Limitations](#) for more information.

Procedure

To create a backup of persistent volumes:

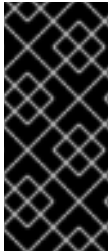
1. Stop the application that is using the persistent volume.
2. Clone the persistent volume.
3. Restart the application.
4. Create a backup of the cloned volume.
5. Delete the cloned volume.

12.3.16. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- [Set up your registry and configure registry storage](#).

12.4. INSTALLING A CLUSTER ON VSPHERE WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.5, you can install a cluster on VMware vSphere infrastructure that you provision.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the vSphere platform and the installation process of OpenShift Container Platform. Use the user-provisioned infrastructure installation instructions as a guide; you are free to create the required resources through other methods.

12.4.1. Prerequisites

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to allow the sites](#) that your cluster requires access to.



NOTE

Be sure to also review this site list if you are configuring a proxy.

12.4.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

12.4.3. VMware vSphere infrastructure requirements

You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.18. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.4.4. Machine requirements for a cluster with user-provisioned infrastructure

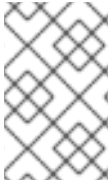
For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

12.4.4.1. Required machines

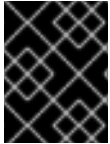
The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine

- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines.

**NOTE**

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

**IMPORTANT**

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

**IMPORTANT**

All virtual machines must reside in the same datastore and in the same folder as the installer.

12.4.4.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

12.4.4.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: $(\text{threads per core} \times \text{cores}) \times \text{sockets} = \text{vCPUs}$.

12.4.4.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

12.4.5. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

12.4.5.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **inittamfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 12.19. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .

Protocol	Port	Description
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 12.20. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 12.21. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

- Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
- A stateless load balancing algorithm. The options vary based on the load balancer implementation.

**NOTE**

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 12.22. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 12.23. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Ethernet adaptor hardware address requirements

When provisioning VMs for the cluster, the ethernet interfaces configured for each VM must use a MAC address from the VMware Organizationally Unique Identifier (OUI) allocation ranges:

- **00:05:69:00:00:00 to 00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00 to 00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00 to 00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00 to 00:50:56:FF:FF:FF**

If a MAC address outside the VMware OUI is used, the cluster installation will not succeed.

Additional resources


- [Configuring chrony time service](#)

12.4.5.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 12.24. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.  IMPORTANT The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	*.apps.<cluster_name>.<base_domain>	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 12.7. Sample DNS zone database

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 12.8. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN

```

```

; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

12.4.6. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```

$ ssh-keygen -t ed25519 -N "" \
  -f <path>/<file_name> 1

```

1

Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

12.4.7. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.

- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.



IMPORTANT

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.



IMPORTANT

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

12.4.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

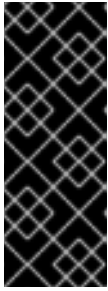
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

**IMPORTANT**

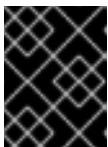
You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

**NOTE**

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

12.4.8.1. Sample install-config.yaml file for VMware vSphere

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
  fips: false 15
  pullSecret: '{"auths": ...}' 16
  sshKey: 'ssh-ed25519 AAAA...' 17

```


- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous multithreading in some cluster machines, you must disable it in all cluster machines.



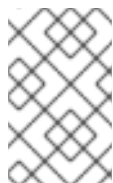
IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Your machines must use at least 8 CPUs and 32 GB of RAM if you disable simultaneous multithreading.

- 4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.
- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8 The cluster name that you specified in your DNS records.
- 9 The fully-qualified host name or IP address of the vCenter server.
- 10 The name of the user for accessing the server. This user must have at least the roles and privileges that are required for [static or dynamic persistent volume provisioning](#) in vSphere.
- 11 The password associated with the vSphere user.
- 12 The vSphere datacenter.
- 13 The default vSphere datastore to use.
- 14 Optional: For installer-provisioned infrastructure, the absolute path of an existing folder where the installation program creates the virtual machines, for example, `/<datacenter_name>/vm/<folder_name>/<subfolder_name>`. If you do not provide this value, the installation program creates a top-level folder in the datacenter virtual machine folder that is named with the infrastructure ID. If you are providing the infrastructure for the cluster, omit this parameter.
- 15 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 16 The pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the

manager site. This `pullSecret` allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

- 17** The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

12.4.8.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.
- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
```

- 1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- 2 A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- 3 A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations. You must include vCenter's IP address and the IP range that you use for its machines.
- 4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

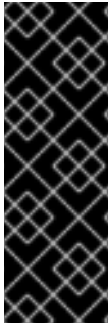
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

12.4.9. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory  
WARNING Making control-plane schedulable by setting MastersSchedulable to true for  
Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

4. Remove the Kubernetes manifest files that define the control plane machines and compute machine sets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the machine set files to create compute machines by using the machine API, but you must update references to them to match your environment.
5. Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - a. Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` file.
 - b. Locate the `mastersSchedulable` parameter and set its value to **False**.
 - c. Save and exit the file.
 6. Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yaml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- 1 2 Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

7. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1 For `<installation_directory>`, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

■

12.4.10. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in VMware vSphere (vSphere). The provided {cp-template} templates contain references to this infrastructure name, so you must extract it.

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in VMware vSphere. If you plan to use the cluster identifier as the name of your virtual machine folder, you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

12.4.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

Before you install a cluster that contains user-provisioned infrastructure on VMware vSphere, you must create RHCOS machines on vSphere hosts for it to use.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.
- Create a [vSphere cluster](#).

Procedure

1. Upload the bootstrap Ignition config file, which is named **<installation_directory>/bootstrap.ign**, that the installation program created to your HTTP server. Note the URL of this file.
You must host the bootstrap Ignition config file because it is too large to fit in a vApp property.
2. Save the following secondary Ignition config file for your bootstrap node to your computer as **<installation_directory>/append-bootstrap.ign**.

```
{
  "ignition": {
    "config": {
      "append": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "2.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

- 1 Specify the URL of the bootstrap Ignition config file that you hosted.

When you create the virtual machine (VM) for the bootstrap machine, you use this Ignition config file.

3. Convert the master, worker, and secondary bootstrap Ignition config files to base64 encoding. For example, if you use a Linux operating system, you can use the **base64** command to encode the files.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

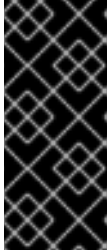
```
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64
```



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

4. Obtain the RHCOS OVA image. Images are available from the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The filename contains the OpenShift Container Platform version number in the format **rhcos-vmware.<architecture>.ova**.

5. In the vSphere Client, create a folder in your datacenter to store your VMs.
 - a. Click the **VMs and Templates** view.
 - b. Right-click the name of your datacenter.
 - c. Click **New Folder → New VM and Template Folder**.
 - d. In the window that is displayed, enter the folder name. If you did not specify an existing folder in the **install-config.yaml** file, create a folder with the same name as the infrastructure ID.
6. In the vSphere Client, create a template for the OVA image and then clone the template as needed.



NOTE

In the following steps, you create a template and then clone the template for all of your cluster machines. You then provide the location for the Ignition config file for that cloned machine type when you provision the VMs.

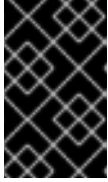
- a. From the **Hosts and Clusters** tab, right-click your cluster name and select **Deploy OVF Template**.
- b. On the **Select an OVF** tab, specify the name of the RHCOS OVA file that you downloaded.
- c. On the **Select a name and folder** tab, set a **Virtual machine name** for your template, such as **Template-RHCOS**. Click the name of your vSphere cluster and select the folder you created in the previous step.
- d. On the **Select a compute resource** tab, click the name of your vSphere cluster.
- e. On the **Select storage** tab, configure the storage options for your VM.
 - Select **Thin Provision** or **Thick Provision**, based on your storage preferences.
 - Select the datastore that you specified in your **install-config.yaml** file.
- f. On the **Select network** tab, specify the network that you configured for the cluster, if available.
- g. When creating the OVF template, do not specify values on the **Customize template** tab or configure the template any further.



IMPORTANT

Do not start the original VM template. The VM template must remain off and must be cloned for new RHCOS machines. Starting the VM template configures the VM template as a VM on the platform, which prevents it from being used as a template that machine sets can apply configurations to.

7. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **control-plane-0** or **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.
 - d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - Optional: In the event of cluster performance issues, from the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
 - **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
 - Alternatively, prior to powering on the virtual machine add via vApp properties:
 - Navigate to a virtual machine from the vCenter Server inventory.
 - On the **Configure** tab, expand **Settings** and select **vApp options**.
 - Scroll down and under **Properties** apply the configurations from above.
 - h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type.
 - i. Complete the configuration and power on the VM.
8. Create the rest of the machines for your cluster by following the preceding steps for each machine.



IMPORTANT

You must create the bootstrap and control plane machines at this time. Because some pods are deployed on compute machines by default, also create at least two compute machines before you install the cluster.

12.4.12. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

You can create more compute machines for your cluster that uses user-provisioned infrastructure on VMware vSphere.

Prerequisites

- Obtain the base64-encoded Ignition file for your compute machines.
- You have access to the vSphere template that you created for your cluster.

Procedure

1. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.
 - d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - From the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
 - **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded compute Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
 - h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type. Also, make sure to select the correct network under **Add network adapter** if there are multiple networks available.
 - i. Complete the configuration and power on the VM.

2. Continue to create more compute machines for your cluster.

12.4.13. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

12.4.13.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.4.13.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.

5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

12.4.13.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.4.14. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
--log-level=info 2
```

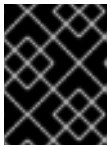
- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

12.4.15. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

12.4.16. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

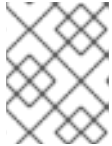
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   73m   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

12.4.17. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s

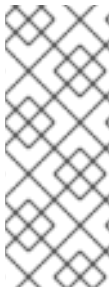
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

12.4.17.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

12.4.17.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

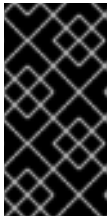
12.4.17.2.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.

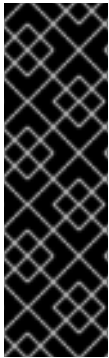
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.



IMPORTANT

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.



NOTE

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

12.4.17.2.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```



WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

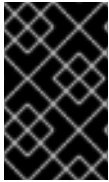
```
managementState: Removed
```

to

```
managementState: Managed
```

12.4.17.2.3. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
 - a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
    - ReadWriteOnce 2
  resources:
    requests:
      storage: 100Gi 3
```

- 1 A unique name that represents the **PersistentVolumeClaim** object.
- 2 The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.
- 3 The size of the PersistentVolumeClaim.

- b. Create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1 Creating a custom PVC allows you to leave the **claim** field blank for the default automatic

creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see [Configuring the registry for vSphere](#).

12.4.18. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m

service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE          NAME                                     READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running    1    9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8  1/1
Running    0    5m
...
```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> 1
```

- 1 Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

You can add extra compute machines after the cluster installation is completed by following [Adding compute machines to vSphere](#).

12.4.19. Backing up VMware vSphere volumes

OpenShift Container Platform provisions new volumes as independent persistent disks to freely attach and detach the volume on any node in the cluster. As a consequence, it is not possible to back up volumes that use snapshots, or to restore volumes from snapshots. See [Snapshot Limitations](#) for more information.

Procedure

To create a backup of persistent volumes:

1. Stop the application that is using the persistent volume.
2. Clone the persistent volume.
3. Restart the application.
4. Create a backup of the cloned volume.
5. Delete the cloned volume.

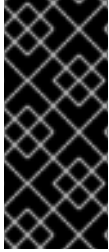
12.4.20. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- [Set up your registry and configure registry storage](#).

12.5. INSTALLING A CLUSTER ON VSPHERE WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.5, you can install a cluster on VMware vSphere infrastructure that you provision with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the vSphere platform and the installation process of OpenShift Container Platform. Use the user-provisioned infrastructure installation instructions as a guide; you are free to create the required resources through other methods.

12.5.1. Prerequisites

- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall, you must [configure it to access Red Hat Insights](#) .

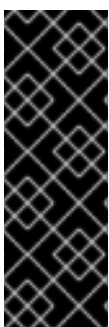
12.5.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#) .

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

12.5.3. VMware vSphere infrastructure requirements

You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.25. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.5.4. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

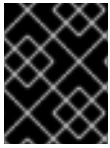
12.5.4.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines.

**NOTE**

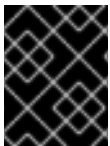
The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

**IMPORTANT**

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

**IMPORTANT**

All virtual machines must reside in the same datastore and in the same folder as the installer.

12.5.4.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

12.5.4.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

- 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

12.5.4.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

12.5.5. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

12.5.5.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 12.26. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn

Protocol	Port	Description
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 12.27. All machines to control plane

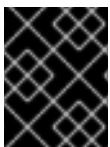
Protocol	Port	Description
TCP	6443	Kubernetes API

Table 12.28. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



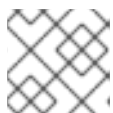
IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 12.29. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 12.30. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Ethernet adaptor hardware address requirements

When provisioning VMs for the cluster, the ethernet interfaces configured for each VM must use a MAC address from the VMware Organizationally Unique Identifier (OUI) allocation ranges:

- **00:05:69:00:00:00 to 00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00 to 00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00 to 00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00 to 00:50:56:FF:FF:FF**

If a MAC address outside the VMware OUI is used, the cluster installation will not succeed.

Additional resources

- [Configuring chrony time service](#)


12.5.5.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>.**

Table 12.31. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

Component	Record	Description
	api-int.<cluster_name>.<base_domain>.	<p>Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.</p> </div> </div>
Routes	*.apps.<cluster_name>.<base_domain>.	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>.	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 12.9. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H ; refresh (3 hours)
  30M ; retry (30 minutes)
```

```

2W ; expiry (2 weeks)
1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 12.10. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
2019070700 ; serial
3H ; refresh (3 hours)
30M ; retry (30 minutes)
2W ; expiry (2 weeks)
1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;

```



```
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

12.5.6. Generating an SSH private key and adding it to the agent

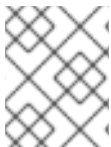
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

-

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

12.5.7. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

- You must install the cluster from a computer that uses Linux or macOS.
- You need 500 MB of local disk space to download the installation program.

Procedure

1. Access the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
2. Navigate to the page for your installation type, download the installation program for your operating system, and place the file in the directory where you will store the installation configuration files.

**IMPORTANT**

The installation program creates several files on the computer that you use to install your cluster. You must keep both the installation program and the files that the installation program creates after you finish installing the cluster.

**IMPORTANT**

Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. You must complete the OpenShift Container Platform uninstallation procedures outlined for your specific cloud provider to remove your cluster entirely.

3. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar xvf <installation_program>.tar.gz
```

4. From the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site, download your installation pull secret as a **.txt** file. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

12.5.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

**IMPORTANT**

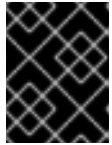
You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.

**NOTE**

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

**IMPORTANT**

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

12.5.8.1. Sample install-config.yaml file for VMware vSphere

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
  name: worker
  replicas: 0 4
controlPlane:
  hyperthreading: Enabled 5 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
platform:
  vsphere:
    vcenter: your.vcenter.server 9
    username: username 10
    password: password 11
    datacenter: datacenter 12
    defaultDatastore: datastore 13
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" 14
  fips: false 15
  pullSecret: '{"auths": ...}' 16
  sshKey: 'ssh-ed25519 AAAA...' 17

```

- 1 The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- 2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, -, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.
- 3 6 Whether to enable or disable simultaneous multithreading, or **hyperthreading**. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable simultaneous

multithreading in some cluster machines, you must disable it in all cluster machines.



IMPORTANT

If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. Your machines must use at least 8 CPUs and 32 GB of RAM if you disable simultaneous multithreading.

- 4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.
- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8 The cluster name that you specified in your DNS records.
- 9 The fully-qualified host name or IP address of the vCenter server.
- 10 The name of the user for accessing the server. This user must have at least the roles and privileges that are required for [static or dynamic persistent volume provisioning](#) in vSphere.
- 11 The password associated with the vSphere user.
- 12 The vSphere datacenter.
- 13 The default vSphere datastore to use.
- 14 Optional: For installer-provisioned infrastructure, the absolute path of an existing folder where the installation program creates the virtual machines, for example, `/<datacenter_name>/vm/<folder_name>/<subfolder_name>`. If you do not provide this value, the installation program creates a top-level folder in the datacenter virtual machine folder that is named with the infrastructure ID. If you are providing the infrastructure for the cluster, omit this parameter.
- 15 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 16 The pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.
- 17 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

12.5.8.2. Network configuration parameters

You can modify your cluster network configuration parameters in the **install-config.yaml** configuration file. The following table describes the parameters.



NOTE

You cannot modify these parameters in the **install-config.yaml** file after installation.

Table 12.32. Required network parameters

Parameter	Description	Value
networking.networkType	The default Container Network Interface (CNI) network provider plug-in to deploy. The OpenShiftSDN plug-in is the only plug-in supported in OpenShift Container Platform 4.5.	The default value is OpenShiftSDN .
networking.clusterNetwork[].cidr	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is 10.128.0.0/14 .
networking.clusterNetwork[].hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 , then each node is assigned a /23 subnet out of the given cidr , allowing for 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .
networking.serviceNetwork[]	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is 172.30.0.0/16 .
networking.machineNetwork[].cidr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is 10.0.0.0/16 .

12.5.9. Modifying advanced network configuration parameters

You can modify the advanced network configuration parameters only before you install the cluster. Advanced configuration customization lets you integrate your cluster into your existing network environment by specifying an MTU or VXLAN port, by allowing customization of **kube-proxy** settings, and by specifying a different **mode** for the **openshiftSDNConfig** parameter.



IMPORTANT

Modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.
- Create the Ignition config files for your cluster.

Procedure

1. Use the following command to create manifests:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

- 1 For **<installation_directory>**, specify the directory name that contains the **manifests/** directory for your cluster.

After creating the file, several network configuration files are in the **manifests/** directory, as shown:

```
$ ls <installation_directory>/manifests/cluster-network-*
```

Example output

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. Open the **cluster-network-03-config.yml** file in an editor and enter a CR that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
```

```
- 172.30.0.0/16
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

- 1 The parameters for the **spec** parameter are only an example. Specify your configuration for the Cluster Network Operator in the CR.

The CNO provides default values for the parameters in the CR, so you must specify only the parameters that you want to change.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.
5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.
6. Remove the Kubernetes manifest files that define the control plane machines and compute machineSets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-
cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the MachineSet files to create compute machines by using the machine API, but you must update references to them to match your environment.

12.5.10. Cluster Network Operator configuration

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a CR object that is named **cluster**. The CR specifies the parameters for the **Network** API in the **operator.openshift.io** API group.

You can specify the cluster network configuration for your OpenShift Container Platform cluster by setting the parameter values for the **defaultNetwork** parameter in the CNO CR. The following CR displays the default configuration for the CNO and explains both the parameters you can configure and the valid parameter values:

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
```



```

...
kubeProxyConfig: 4
  iptablesSyncPeriod: 30s 5
  proxyArguments:
    iptables-min-sync-period: 6
    - 0s

```

- 1 2 Specified in the **install-config.yaml** file.
- 3 Configures the default Container Network Interface (CNI) network provider for the cluster network.
- 4 The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.
- 5 The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#) documentation.



NOTE

Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

- 6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the [Go time package](#).

12.5.10.1. Configuration parameters for the OpenShift SDN default CNI network provider

The following YAML object describes the configuration parameters for the OpenShift SDN default Container Network Interface (CNI) network provider.

```

defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5

```

- 1 Specified in the **install-config.yaml** file.
- 2 Specify only if you want to override part of the OpenShift SDN configuration.
- 3 Configures the network isolation mode for OpenShift SDN. The allowed values are **Multitenant**, **Subnet**, or **NetworkPolicy**. The default value is **NetworkPolicy**.
- 4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

If the auto-detected value is not what you expected it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.

If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1450**.

- 5** The port to use for all VXLAN packets. The default value is **4789**. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for VXLAN, since both SDNs use the same default VXLAN port number.

On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

12.5.10.2. Cluster Network Operator example configuration

A complete CR object for the CNO is displayed in the following example:

Cluster Network Operator example CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
        - 0s
```

12.5.11. Creating the Ignition config files

Because you must manually start the cluster machines, you must generate the Ignition config files that the cluster needs to make its machines.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

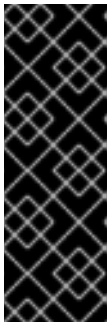
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

- Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

If you created an **install-config.yaml** file, specify the directory that contains it. Otherwise, specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.5.12. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in `{cp-first}` (`{cp}`). The provided `{cp-template}` templates contain references to this infrastructure name, so you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

12.5.13. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

Before you install a cluster that contains user-provisioned infrastructure on VMware vSphere, you must create RHCOS machines on vSphere hosts for it to use.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.
- Create a [vSphere cluster](#).

Procedure

1. Upload the bootstrap Ignition config file, which is named **<installation_directory>/bootstrap.ign**, that the installation program created to your HTTP server. Note the URL of this file.
You must host the bootstrap Ignition config file because it is too large to fit in a vApp property.
2. Save the following secondary Ignition config file for your bootstrap node to your computer as **<installation_directory>/append-bootstrap.ign**.

```
{
  "ignition": {
    "config": {
      "append": [
        {
```

```

    "source": "<bootstrap_ignition_config_url>", ❶
    "verification": {}
  }
]
},
"timeouts": {},
"version": "2.2.0"
},
"networkd": {},
"passwd": {},
"storage": {},
"systemd": {}
}

```

- ❶ Specify the URL of the bootstrap Ignition config file that you hosted.

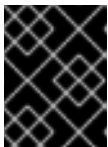
When you create the virtual machine (VM) for the bootstrap machine, you use this Ignition config file.

- Convert the master, worker, and secondary bootstrap Ignition config files to base64 encoding. For example, if you use a Linux operating system, you can use the **base64** command to encode the files.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64
```



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Obtain the RHCOS OVA image. Images are available from the [RHCOS image mirror](#) page.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The filename contains the OpenShift Container Platform version number in the format **rhcos-vmware.<architecture>.ova**.

- In the vSphere Client, create a folder in your datacenter to store your VMs.
 - Click the **VMs and Templates** view.

- b. Right-click the name of your datacenter.
 - c. Click **New Folder → New VM and Template Folder**.
 - d. In the window that is displayed, enter the folder name. If you did not specify an existing folder in the **install-config.yaml** file, create a folder with the same name as the infrastructure ID.
6. In the vSphere Client, create a template for the OVA image and then clone the template as needed.



NOTE

In the following steps, you create a template and then clone the template for all of your cluster machines. You then provide the location for the Ignition config file for that cloned machine type when you provision the VMs.

- a. From the **Hosts and Clusters** tab, right-click your cluster name and select **Deploy OVF Template**.
- b. On the **Select an OVF** tab, specify the name of the RHCOS OVA file that you downloaded.
- c. On the **Select a name and folder** tab, set a **Virtual machine name** for your template, such as **Template-RHCOS**. Click the name of your vSphere cluster and select the folder you created in the previous step.
- d. On the **Select a compute resource** tab, click the name of your vSphere cluster.
- e. On the **Select storage** tab, configure the storage options for your VM.
 - Select **Thin Provision** or **Thick Provision**, based on your storage preferences.
 - Select the datastore that you specified in your **install-config.yaml** file.
- f. On the **Select network** tab, specify the network that you configured for the cluster, if available.
- g. When creating the OVF template, do not specify values on the **Customize template** tab or configure the template any further.



IMPORTANT

Do not start the original VM template. The VM template must remain off and must be cloned for new RHCOS machines. Starting the VM template configures the VM template as a VM on the platform, which prevents it from being used as a template that machine sets can apply configurations to.

7. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**.
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **control-plane-0** or **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.

- d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - Optional: In the event of cluster performance issues, from the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
 - **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
 - Alternatively, prior to powering on the virtual machine add via vApp properties:
 - Navigate to a virtual machine from the vCenter Server inventory.
 - On the **Configure** tab, expand **Settings** and select **vApp options**.
 - Scroll down and under **Properties** apply the configurations from above.
 - h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type.
 - i. Complete the configuration and power on the VM.
8. Create the rest of the machines for your cluster by following the preceding steps for each machine.



IMPORTANT

You must create the bootstrap and control plane machines at this time. Because some pods are deployed on compute machines by default, also create at least two compute machines before you install the cluster.

12.5.14. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

You can create more compute machines for your cluster that uses user-provisioned infrastructure on VMware vSphere.

Prerequisites

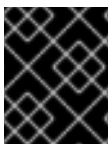
- Obtain the base64-encoded Ignition file for your compute machines.
- You have access to the vSphere template that you created for your cluster.

Procedure

1. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.
 - d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - From the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
 - **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded compute Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
 - h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type. Also, make sure to select the correct network under **Add network adapter** if there are multiple networks available.
 - i. Complete the configuration and power on the VM.
2. Continue to create more compute machines for your cluster.

12.5.15. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

12.5.15.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**.
To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.5.15.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

12.5.15.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.

3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.5.16. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1  
--log-level=info 2
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...  
INFO API v1.18.3 up  
INFO Waiting up to 30m0s for bootstrapping to complete...  
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

12.5.17. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

12.5.18. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```

NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   63m   v1.18.3
master-1  Ready    master   63m   v1.18.3
master-2  Ready    master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3

```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```

NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps 15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...

```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}\n{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS  ROLES  AGE  VERSION
master-0  Ready   master 73m  v1.20.0
master-1  Ready   master 73m  v1.20.0
master-2  Ready   master 74m  v1.20.0
worker-0  Ready   worker 11m  v1.20.0
worker-1  Ready   worker 11m  v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

12.5.19. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	69s
cloud-credential	4.5.4	True	False	False	12m
cluster-autoscaler	4.5.4	True	False	False	11m
console	4.5.4	True	False	False	46s
dns	4.5.4	True	False	False	11m
image-registry	4.5.4	True	False	False	5m26s
ingress	4.5.4	True	False	False	5m36s
kube-apiserver	4.5.4	True	False	False	8m53s
kube-controller-manager	4.5.4	True	False	False	7m24s
kube-scheduler	4.5.4	True	False	False	12m
machine-api	4.5.4	True	False	False	12m
machine-config	4.5.4	True	False	False	7m36s
marketplace	4.5.4	True	False	False	7m54m
monitoring	4.5.4	True	False	False	7h54s
network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

12.5.19.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

12.5.19.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

12.5.19.2.1. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



IMPORTANT

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
 - a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
    - ReadWriteOnce 2
```

```
resources:
  requests:
    storage: 100Gi 3
```

- 1** A unique name that represents the **PersistentVolumeClaim** object.
- 2** The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.
- 3** The size of the PersistentVolumeClaim.

b. Create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Creating a custom PVC allows you to leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see [Configuring the registry for vSphere](#).

12.5.20. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME                                VERSION AVAILABLE PROGRESSING DEGRADED
```


SINCE					
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

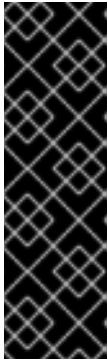
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.
 - a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE           NAME                                     READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running   1    9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator  authentication-operator-69d5d8bf84-vh2n8      1/1
Running   0    5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> ❶
```

- ❶ Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

You can add extra compute machines after the cluster installation is completed by following [Adding compute machines to vSphere](#).

12.5.21. Backing up VMware vSphere volumes

OpenShift Container Platform provisions new volumes as independent persistent disks to freely attach and detach the volume on any node in the cluster. As a consequence, it is not possible to back up volumes that use snapshots, or to restore volumes from snapshots. See [Snapshot Limitations](#) for more information.

Procedure

To create a backup of persistent volumes:

1. Stop the application that is using the persistent volume.
2. Clone the persistent volume.
3. Restart the application.
4. Create a backup of the cloned volume.
5. Delete the cloned volume.

12.5.22. Next steps

- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- [Set up your registry and configure registry storage](#).

12.6. INSTALLING A CLUSTER ON VSPHERE IN A RESTRICTED NETWORK

In OpenShift Container Platform 4.5, you can install a cluster on VMware vSphere infrastructure in a restricted network by creating an internal mirror of the installation release content.

12.6.1. Prerequisites

- [Create a registry on your mirror host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide the ReadWriteMany access mode.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.



NOTE

If you are configuring a proxy, be sure to also review this site list.

12.6.2. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be

completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.

12.6.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

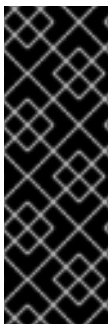
12.6.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

12.6.4. VMware vSphere infrastructure requirements

You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.33. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.6.5. vCenter requirements

Before you install an OpenShift Container Platform cluster on your vCenter that uses infrastructure that the installer provisions, you must prepare your environment.

Required vCenter account privileges

To install an OpenShift Container Platform cluster in a vCenter, the installation program requires access to an account with privileges to read and create the required resources. Using an account that has global administrative privileges is the simplest way to access all of the necessary permissions.

If you cannot use an account with global administrative privileges, you must create roles to grant the privileges necessary for OpenShift Container Platform cluster installation. While most of the privileges are always required, some are required only if you plan for the installation program to provision a folder to contain the OpenShift Container Platform cluster on your vCenter instance, which is the default behavior. You must create or amend vSphere roles for the specified objects to grant the required privileges.

An additional role is required if the installation program is to create a vSphere virtual machine folder.

Example 12.11. Roles and privileges required for installation

vSphere object for role	When required	Required privileges
vSphere vCenter	Always	Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View
vSphere vCenter Cluster	Always	Host.Config.StorageResource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk
vSphere Datastore	Always	Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement
vSphere Port Group	Always	Network.Assign

vSphere object for role	When required	Required privileges
Virtual Machine Folder	Always	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone

vSphere object for role	When required	Required privileges
vSphere vCenter Datacenter	If the installation program creates the virtual machine folder	Resource.AssignVMToPool VApp.Import VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.Annotation VirtualMachine.Config.CPUCount VirtualMachine.Config.DiskExtend VirtualMachine.Config.DiskLease VirtualMachine.Config.EditDevice VirtualMachine.Config.Memory VirtualMachine.Config.RemoveDisk VirtualMachine.Config.Rename VirtualMachine.Config.ResetGuestInfo VirtualMachine.Config.Resource VirtualMachine.Config.Settings VirtualMachine.Config.UpgradeVirtualHardware VirtualMachine.Interact.GuestControl VirtualMachine.Interact.PowerOff VirtualMachine.Interact.PowerOn VirtualMachine.Interact.Reset VirtualMachine.Inventory.Create VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.Delete VirtualMachine.Provisioning.Clone Folder.Create Folder.Delete

Additionally, the user requires some **ReadOnly** permissions, and some of the roles require permission to propagate the permissions to child objects. These settings vary depending on whether or not you install the cluster into an existing folder.

Example 12.12. Required permissions and propagation settings

vSphere object	Folder type	Propagate to children	Permissions required
vSphere vCenter	Always	False	Listed required privileges
vSphere vCenter Datacenter	Existing folder	False	ReadOnly permission
	Installation program creates the folder	True	Listed required privileges
vSphere vCenter Cluster	Always	True	Listed required privileges
vSphere vCenter Datastore	Always	False	Listed required privileges
vSphere Switch	Always	False	ReadOnly permission
vSphere Port Group	Always	False	Listed required privileges
vSphere vCenter Virtual Machine Folder	Existing folder	True	Listed required privileges

For more information about creating an account with only the required privileges, see [vSphere Permissions and User Management Tasks](#) in the vSphere documentation.

Using OpenShift Container Platform with vMotion



IMPORTANT

OpenShift Container Platform generally supports compute-only vMotion. Using Storage vMotion can cause issues and is not supported.

If you are using vSphere volumes in your pods, migrating a VM across datastores either manually or through Storage vMotion causes invalid references within OpenShift Container Platform persistent volume (PV) objects. These references prevent affected pods from starting up and can result in data loss.

Similarly, OpenShift Container Platform does not support selective migration of VMDKs across datastores, using datastore clusters for VM provisioning or for dynamic or static provisioning of PVs, or using a datastore that is part of a datastore cluster for dynamic or static provisioning of PVs.

Cluster resources

When you deploy an OpenShift Container Platform cluster that uses installer-provisioned infrastructure, the installation program must be able to create several resources in your vCenter instance.

A standard OpenShift Container Platform installation creates the following vCenter resources:

- 1 Folder
- 1 Tag category
- 1 Tag
- Virtual machines:
 - 1 template
 - 1 temporary bootstrap node
 - 3 control plane nodes
 - 3 compute machines

Although these resources use 856 GB of storage, the bootstrap node is destroyed during the cluster installation process. A minimum of 800 GB of storage is required to use a standard cluster.

If you deploy more compute machines, the OpenShift Container Platform cluster will use more storage.

Cluster limits

Available resources vary between clusters. The number of possible clusters within a vCenter is limited primarily by available storage space and any limitations on the number of required resources. Be sure to consider both limitations to the vCenter resources that the cluster creates and the resources that you require to deploy a cluster, such as IP addresses and networks.

Networking requirements

You must use DHCP for the network and ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines. The VM in your restricted network must have access to vCenter so that it can provision and manage nodes, persistent volume claims (PVCs), and other resources. Additionally, you must create the following networking resources before you install the OpenShift Container Platform cluster:

Required IP addresses

An installer-provisioned vSphere installation requires two static IP addresses:

- The **API** address is used to access the cluster API.
- The **Ingress** address is used for cluster ingress traffic.

You must provide these IP addresses to the installation program when you install the OpenShift Container Platform cluster.

DNS records

You must create DNS records for two static IP addresses in the appropriate DNS server for the vCenter instance that hosts your OpenShift Container Platform cluster. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify when you install the

cluster. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>.**

Table 12.34. Required DNS records

Component	Record	Description
API VIP	api.<cluster_name>.<base_domain>.	This DNS A/AAAA or CNAME record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Ingress VIP	*.apps.<cluster_name>.<base_domain>.	A wildcard DNS A/AAAA or CNAME record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

12.6.6. Generating an SSH private key and adding it to the agent

If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> 1
```

- 1 Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`
2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

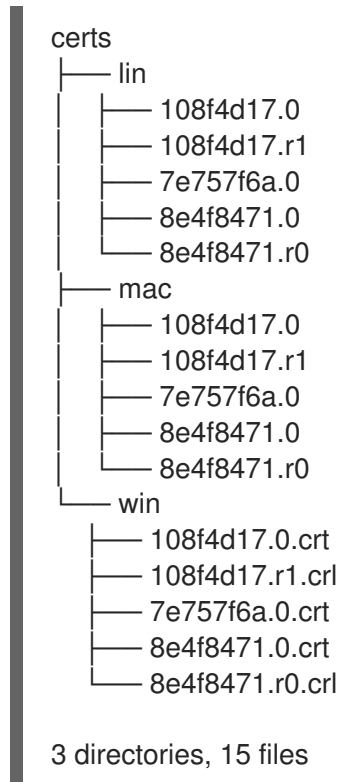
- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

12.6.7. Adding vCenter root CA certificates to your system trust

Because the installation program requires access to your vCenter's API, you must add your vCenter's trusted root CA certificates to your system trust before you install an OpenShift Container Platform cluster.

Procedure

1. From the vCenter home page, download the vCenter's root CA certificates. Click **Download trusted root CA certificates** in the vSphere Web Services SDK section. The **<vCenter>/certs/download.zip** file downloads.
2. Extract the compressed file that contains the vCenter root CA certificates. The contents of the compressed file resemble the following file structure:



3. Add the files for your operating system to the system trust. For example, on a Fedora operating system, run the following command:

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. Update your system trust. For example, on a Fedora operating system, run the following command:

```
# update-ca-trust extract
```

12.6.8. Creating the RHCOS image for restricted network installations

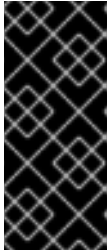
Download the Red Hat Enterprise Linux CoreOS (RHCOS) image to install OpenShift Container Platform on a restricted network VMware vSphere environment.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, the program is on your mirror registry host.

Procedure

1. Log in to the Red Hat Customer Portal's [Product Downloads](#) page.
2. Under **Version**, select the most recent release of OpenShift Container Platform 4.5 for RHEL 8.



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available.

3. Download the **Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere** image.
4. Upload the image you downloaded to a location that is accessible from the bastion server.

The image is now available for a restricted installation. Note the image name or location for use in OpenShift Container Platform deployment.

12.6.9. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Google Cloud Platform (GCP). VMware vSphere.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your bastion host.
- Have the **imageContentSources** values that were generated during mirror registry creation.
- Obtain the contents of the certificate for your mirror registry.

Procedure

1. Create the **install-config.yaml** file.
 - a. Run the following command:

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

- 1** For **<installation_directory>**, specify the directory name to store the files that the installation program creates.



IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

- b. At the prompts, provide the configuration details for your cloud:
 - i. Optional: Select an SSH key to use to access your cluster machines.



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- ii. Select **gcp** as the platform to target.
- iii. If you have not configured the service account key for your GCP account on your computer, you must obtain it from GCP and paste the contents of the file or enter the absolute path to the file.
- iv. Select the project ID to provision the cluster in. The default value is specified by the service account that you configured.
- v. Select the region to deploy the cluster to.
- vi. Select the base domain to deploy the cluster to. The base domain corresponds to the public DNS zone that you created for your cluster.
- vii. Select **vsphere** as the platform to target.
- viii. Specify the name of your vCenter instance.
- ix. Specify the user name and password for the vCenter account that has the required permissions to create the cluster.
The installation program connects to your vCenter instance.
- x. Select the datacenter in your vCenter instance to connect to.
- xi. Select the default vCenter datastore to use.
- xii. Select the vCenter cluster to install the OpenShift Container Platform cluster in.
- xiii. Select the network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.
- xiv. Enter the virtual IP address that you configured for control plane API access.
- xv. Enter the virtual IP address that you configured for cluster ingress.

- xvi. Enter the base domain. This base domain must be the same one that you used in the DNS records that you configured.
 - xvii. Enter a descriptive name for your cluster. The cluster name must be the same one that you used in the DNS records that you configured.
 - xviii. Paste the pull secret that you obtained from the [Pull Secret](#) page on the Red Hat OpenShift Cluster Manager site.
2. In the **install-config.yaml** file, set the value of **platform.vsphere.clusterOSImage** to the image location or name. For example:

```
platform:
  vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-vmware.x86_64.ova?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. Edit the **install-config.yaml** file to provide the additional information that is required for an installation in a restricted network.
- a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<bastion_host_name>:5000":{"auth": "<credentials>","email": "you@example.com"}}}'
```

For **<bastion_host_name>**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

- b. Add the **additionalTrustBundle** parameter and value.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  /-----END CERTIFICATE-----
```

The value must be the contents of the certificate file that you used for your mirror registry, which can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

- c. Define the network and subnets for the VPC to install the cluster in under the parent **platform.gcp** field:

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

For **platform.gcp.network**, specify the name for the existing Google VPC. For **platform.gcp.controlPlaneSubnet** and **platform.gcp.computeSubnet**, specify the existing subnets to deploy the control plane machines and compute machines, respectively.

- d. Add the image content resources, which look like this excerpt:

```
imageContentSources:
```


- mirrors:
 - <bastion_host_name>:5000/<repo_name>/release
 - source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
 - <bastion_host_name>:5000/<repo_name>/release
 - source: registry.example.com/ocp/release

To complete these values, use the **imageContentSources** that you recorded during mirror registry creation.

4. Make any other modifications to the **install-config.yaml** file that you require. You can find more information about the available parameters in the **Installation configuration parameters** section.
5. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

12.6.9.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.



NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.



IMPORTANT

The **openshift-install** command does not validate field names for parameters. If an incorrect name is specified, the related file or object is not created, and no error is reported. Ensure that the field names for any parameters that are specified are correct.

12.6.9.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 12.35. Required parameters

Parameter	Description	Values
apiVersion	The API version for the install-config.yaml content. The current version is v1 . The installer may also support older API versions.	String


Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the baseDomain and metadata.name parameter values that uses the <metadata.name>.<baseDomain> format.	A fully-qualified domain or subdomain name, such as example.com .
metadata	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform	The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere . For additional information about platform.<platform> parameters, consult the following table for your specific platform.	Object
pullSecret	Get a pull secret from https://cloud.redhat.com/openshift/install/pull-secret to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>


12.6.9.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

Only IPv4 addresses are supported.

Table 12.36. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object  NOTE You cannot modify parameters specified by the networking object after installation.
networking.networkType	The cluster network provider Container Network Interface (CNI) plug-in to install.	Either OpenShiftSDN or OVNKubernetes . The default value is OpenShiftSDN .
networking.clusterNetwork	The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23 . If you specify multiple IP address blocks, the blocks must not overlap.	An array of objects. For example: <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</pre>
networking.clusterNetwork.cidr	Required if you use networking.clusterNetwork . An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32 .
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23 then each node is assigned a /23 subnet out of the given cidr . A hostPrefix value of 23 provides 510 ($2^{(32 - 23)} - 2$) pod IP addresses.	A subnet prefix. The default value is 23 .

Parameter	Description	Values
networking.serviceNetwork	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16</pre>
networking.machineNetwork	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>
networking.machineNetwork.cidr	<p>Required if you use networking.machineNetwork. An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt. For libvirt, the default value is 192.168.126.0/24.</p>	<p>An IP network block in CIDR notation.</p> <p>For example, 10.0.0.0/16.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in.</p> </div> </div>

12.6.9.1.3. Optional configuration parameters



Optional installation configuration parameters are described in the following table:

Table 12.37. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
compute	The configuration for the machines that comprise the compute nodes.	Array of machine-pool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
compute.name	Required if you use compute . The name of the machine pool.	worker
compute.platform	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to 2 . The default value is 3 .
controlPlane	The configuration for the machines that comprise the control plane.	Array of MachinePool objects. For details, see the following "Machine-pool" table.

Parameter	Description	Values
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, heterogeneous clusters are not supported, so all pools must specify the same architecture. Valid values are amd64 (the default).	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or hyperthreading , on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.  IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	Enabled or Disabled
controlPlane.name	Required if you use controlPlane . The name of the machine pool.	master
controlPlane.platform	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	aws, azure, gcp, openstack, ovirt, vsphere , or {}
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is 3 , which is the default value.

Parameter	Description	Values
fips	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>If you are using Azure File storage, you cannot enable FIPS mode.</p> </div> </div>	false or true
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources.source	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	Internal or External . To deploy a private cluster, which cannot be accessed from the internet, set publish to Internal . The default value is External .
sshKey	<p>The SSH key to authenticate access to your cluster machines.</p> <div style="display: flex; align-items: flex-start;">  <div> <p>NOTE</p> <p>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your ssh-agent process uses.</p> </div> </div>	For example, sshKey: ssh-ed25519 AAAA...

12.6.9.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

Table 12.38. Additional GCP parameters

Parameter	Description	Values
platform.gcp.network	The name of the existing VPC that you want to deploy your cluster to.	String.
platform.gcp.type	The GCP machine type .	The GCP machine type.
platform.gcp.zones	The availability zones where the installation program creates machines for the specified MachinePool.	A list of valid GCP availability zones , such as us-central1-a , in a YAML sequence .
platform.gcp.controlPlaneSubnet	The name of the existing subnet in your VPC that you want to deploy your control plane machines to.	The subnet name.
platform.gcp.computeSubnet	The name of the existing subnet in your VPC that you want to deploy your compute machines to.	The subnet name.

Table 12.39. Additional VMware vSphere cluster parameters

Parameter	Description	Values
platform.vsphere.vCenter	The fully-qualified hostname or IP address of the vCenter server.	String
platform.vsphere.username	The user name to use to connect to the vCenter instance with. This user must have at least the roles and privileges that are required for static or dynamic persistent volume provisioning in vSphere.	String
platform.vsphere.password	The password for the vCenter user name.	String
platform.vsphere.datacenter	The name of the datacenter to use in the vCenter instance.	String
platform.vsphere.defaultDatastore	The name of the default datastore to use for provisioning volumes.	String

Parameter	Description	Values
platform.vsphere.folder	<i>Optional.</i> The absolute path of an existing folder where the installation program creates the virtual machines. If you do not provide this value, the installation program creates a folder that is named with the infrastructure ID in the datacenter virtual machine folder.	String, for example, /<datacenter_name>/vm/<folder_name>/<subfolder_name> .
platform.vsphere.network	The network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.	String
platform.vsphere.cluster	The vCenter cluster to install the OpenShift Container Platform cluster in.	String
platform.vsphere.apiVIP	The virtual IP (VIP) address that you configured for control plane API access.	An IP address, for example 128.0.0.1 .
platform.vsphere.ingressVIP	The virtual IP (VIP) address that you configured for cluster ingress.	An IP address, for example 128.0.0.1 .

12.6.9.1.5. Optional VMware vSphere machine pool configuration parameters

Optional VMware vSphere machine pool configuration parameters are described in the following table:

Table 12.40. Optional VMware vSphere machine pool parameters

Parameter	Description	Values
platform.vsphere.clusterOSImage	The location from which the installer downloads the RHCOS image. You must set this parameter to perform an installation in a restricted network.	An HTTP or HTTPS URL, optionally with a SHA-256 checksum. For example, https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova .
platform.vsphere.osDisk.diskSizeGB	The size of the disk in gigabytes.	Integer
platform.vsphere.cpus	The total number of virtual processor cores to assign a virtual machine.	Integer

Parameter	Description	Values
platform.vsphere.coresPerSocket	The number of cores per socket in a virtual machine. The number of virtual CPUs (vCPUs) on the virtual machine is platform.vsphere.cpus/platform.vsphere.coresPerSocket . The default value is 1	Integer
platform.vsphere.memoryMB	The size of a virtual machine's memory in megabytes.	Integer

12.6.9.2. Sample install-config.yaml file for an installer-provisioned VMware vSphere cluster

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```

apiVersion: v1
baseDomain: example.com ①
compute: ②
- hyperthreading: Enabled ③
  name: worker
  replicas: 3
  platform:
    vsphere: ④
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8196
      osDisk:
        diskSizeGB: 120
controlPlane: ⑤
  hyperthreading: Enabled ⑥
  name: master
  replicas: 3
  platform:
    vsphere: ⑦
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster ⑧
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore

```


- 11 Provide the contents of the certificate file that you used for your mirror registry.
- 12 Provide the **imageContentSources** section from the output of the command to mirror the repository.

12.6.10. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.



IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

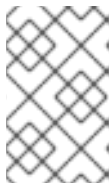
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Remove any existing GCP credentials that do not use the service account key for the GCP account that you configured for your cluster and that are stored in the following locations:
 - The **GOOGLE_CREDENTIALS**, **GOOGLE_CLOUD_KEYFILE_JSON**, or **GKLOUD_KEYFILE_JSON** environment variables
 - The `~/.gcp/osServiceAccount.json` file
 - The **gcloud cli** default credentials
2. Run the installation program:

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
--log-level=info 2
```

- 1 For **<installation_directory>**, specify the location of your customized `./install-config.yaml` file.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

If the cloud provider account that you configured on your host does not have sufficient permissions to deploy the cluster, the installation process stops, and the missing permissions are displayed.

When the cluster deployment completes, directions for accessing your cluster, including a link to its web console and credentials for the **kubeadmin** user, display in your terminal.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

**IMPORTANT**

You must not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

3. Optional: You can reduce the number of permissions for the service account that you used to install the cluster.
 - If you assigned the **Owner** role to your service account, you can remove that role and replace it with the **Viewer** role.
 - If you included the **Service Account Key Admin** role, you can remove it.

12.6.11. Installing the CLI by downloading the binary

You can install the OpenShift CLI (**oc**) in order to interact with OpenShift Container Platform from a command-line interface. You can install **oc** on Linux, Windows, or macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.5. Download and install the new version of **oc**.

12.6.11.1. Installing the CLI on Linux

You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Linux** from the drop-down menu and click **Download command-line tools**.
4. Unpack the archive:

```
$ tar xvzf <file>
```

5. Place the **oc** binary in a directory that is on your **PATH**. To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.6.11.2. Installing the CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **Windows** from the drop-down menu and click **Download command-line tools**.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

12.6.11.3. Installing the CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. Navigate to the [Infrastructure Provider](#) page on the Red Hat OpenShift Cluster Manager site.
2. Select your infrastructure provider, and, if applicable, your installation type.
3. In the **Command line interface** section, select **MacOS** from the drop-down menu and click **Download command-line tools**.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the CLI, it is available using the **oc** command:

```
$ oc <command>
```

12.6.12. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

12.6.13. Creating registry storage

After you install the cluster, you must create storage for the Registry Operator.

12.6.13.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.

**NOTE**

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing `configs.imageregistry.operator.openshift.io`."

12.6.13.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

12.6.13.2.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

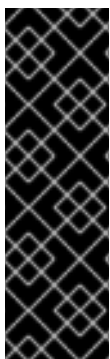
Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

**IMPORTANT**

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.

**IMPORTANT**

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.



NOTE

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

12.6.14. Next steps

- [Customize your cluster.](#)
- If necessary, you can [opt out of remote health reporting](#) .
- [Set up your registry and configure registry storage](#) .

12.7. INSTALLING A CLUSTER ON VSPHERE IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.5, you can install a cluster on VMware vSphere infrastructure that you provision in a restricted network.



IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the vSphere platform and the installation process of OpenShift Container Platform. Use the user-provisioned infrastructure installation instructions as a guide; you are free to create the required resources through other methods.

12.7.1. Prerequisites

- [Create a registry on your mirror host](#) and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- Provision [persistent storage](#) for your cluster. To deploy a private image registry, your storage must provide **ReadWriteMany** access modes.
- Review details about the [OpenShift Container Platform installation and update](#) processes.
- If you use a firewall and plan to use telemetry, you must [configure the firewall to allow the sites](#) that your cluster requires access to.



NOTE

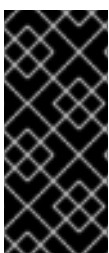
Be sure to also review this site list if you are configuring a proxy.

12.7.2. About installations in restricted networks

In OpenShift Container Platform 4.5, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's IAM service, require Internet access, so you might still require Internet access. Depending on your network, you might require less Internet access for an installation on bare metal hardware or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift Container Platform registry and contains the installation media. You can create this registry on a mirror host, which can access both the Internet and your closed network, or by using other methods that meet your restrictions.



IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

12.7.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

12.7.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.5, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the [Red Hat OpenShift Cluster Manager \(OCM\)](#).

Once you confirm that your Red Hat OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually using OCM, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

You must have Internet access to:

- Access the [Red Hat OpenShift Cluster Manager](#) page to download the installation program and perform subscription management. If the cluster has Internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access [Quay.io](#) to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

If your cluster cannot have direct Internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the content that is required and use it to populate a mirror registry with the packages that you need to install a cluster and generate the installation program. With some installation types, the environment that you install your cluster in will not require Internet access. Before you update the cluster, you update the content of the mirror registry.

12.7.4. VMware vSphere infrastructure requirements

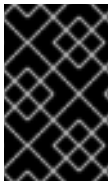
You must install the OpenShift Container Platform cluster on a VMware vSphere version 6 instance that meets the requirements for the components that you use.

Table 12.41. Minimum supported vSphere version for VMware components

Component	Minimum supported versions	Description
-----------	----------------------------	-------------

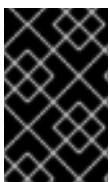
Component	Minimum supported versions	Description
Hypervisor	vSphere 6.5 with HW version 13	This version is the minimum version that Red Hat Enterprise Linux CoreOS (RHCOS) supports. See the Red Hat Enterprise Linux 8 supported hypervisors list .
Networking (NSX-T)	vSphere 6.5U3 or vSphere 6.7U2 and later	vSphere 6.5U3 or vSphere 6.7U2+ are required for OpenShift Container Platform. VMware's NSX Container Plug-in (NCP) 3.0.2 is certified with OpenShift Container Platform 4.5 and NSX-T 3.x+.
Storage with in-tree drivers	vSphere 6.5 and later	This plug-in creates vSphere storage by using the in-tree storage drivers for vSphere included in OpenShift Container Platform.

If you use a vSphere version 6.5 instance, consider upgrading to 6.7U2 before you install OpenShift Container Platform.



IMPORTANT

You must ensure that the time on your ESXi hosts is synchronized before you install OpenShift Container Platform. See [Edit Time Configuration for a Host](#) in the VMware documentation.



IMPORTANT

A limitation of using VPC is that the Storage Distributed Resource Scheduler (SDRS) is not supported. See [vSphere Storage for Kubernetes FAQs](#) in the VMware documentation.

12.7.5. Machine requirements for a cluster with user-provisioned infrastructure

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

12.7.5.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines

- At least two compute machines, which are also known as worker machines.

**NOTE**

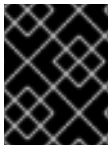
The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

**IMPORTANT**

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See [Red Hat Enterprise Linux technology capabilities and limits](#) .

**IMPORTANT**

All virtual machines must reside in the same datastore and in the same folder as the installer.

12.7.5.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files. Additionally, each OpenShift Container Platform node in the cluster must have access to a Network Time Protocol (NTP) server. If a DHCP server provides NTP servers information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

12.7.5.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU [1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU is equivalent to 1 physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

12.7.5.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

12.7.6. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the [OpenShift Container Platform 4.x Tested Integrations](#) page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

12.7.6.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 12.42. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	1936	Metrics
	9000-9999	Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 .
	10250-10259	The default ports that Kubernetes reserves

Protocol	Port	Description
	10256	openshift-sdn
UDP	4789	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-9999	Host level services, including the node exporter on ports 9100-9101 .
TCP/UDP	30000-32767	Kubernetes node port

Table 12.43. All machines to control plane

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 12.44. Control plane machines to control plane machines

Protocol	Port	Description
TCP	2379-2380	etcd server and peer ports

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



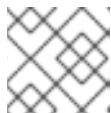
IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer.** Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

**NOTE**

Session persistence is not required for the API load balancer to function properly.

Configure the following ports on both the front and back of the load balancers:

Table 12.45. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

**NOTE**

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

2. **Application Ingress load balancer.** Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:
 - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
 - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

Configure the following ports on both the front and back of the load balancers:

Table 12.46. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTPS traffic

Port	Back-end machines (pool members)	Internal	External	Description
80	The machines that run the Ingress router pods, compute, or worker, by default.	X	X	HTTP traffic

TIP

If the true IP address of the client can be seen by the load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

**NOTE**

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

Ethernet adaptor hardware address requirements

When provisioning VMs for the cluster, the ethernet interfaces configured for each VM must use a MAC address from the VMware Organizationally Unique Identifier (OUI) allocation ranges:

- **00:05:69:00:00:00 to 00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00 to 00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00 to 00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00 to 00:50:56:FF:FF:FF**

If a MAC address outside the VMware OUI is used, the cluster installation will not succeed.

Additional resources


- [Configuring chrony time service](#)

12.7.6.2. User-provisioned DNS requirements

DNS is used for name resolution and reverse name resolution. DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the host name for all the nodes. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 12.47. Required DNS records

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster. <div data-bbox="740 734 844 987" style="display: inline-block; vertical-align: middle;">  </div> <p>IMPORTANT</p> <p>The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.</p>
Routes	*.apps.<cluster_name>.<base_domain>	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	bootstrap.<cluster_name>.<base_domain>	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	<master><n>.<cluster_name>.<base_domain>	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	<worker><n>.<cluster_name>.<base_domain>	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 12.13. Sample DNS zone database

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4 IN A 192.168.1.5
api-int.ocp4 IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4 IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4 IN A 192.168.1.97
master1.ocp4 IN A 192.168.1.98
master2.ocp4 IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4 IN A 192.168.1.11
worker1.ocp4 IN A 192.168.1.7
;
;EOF

```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 12.14. Sample DNS zone database for reverse records

```

$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN

```

```

; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF

```

12.7.7. Generating an SSH private key and adding it to the agent

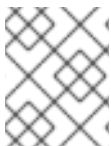
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

In a production environment, you require disaster recovery and debugging.

You can use this key to SSH into the master nodes as the user **core**. When you deploy the cluster, the key is added to the **core** user's `~/.ssh/authorized_keys` list.



NOTE

You must use a local key, not one that you configured with platform-specific approaches such as [AWS key pairs](#).

Procedure

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```

$ ssh-keygen -t ed25519 -N "" \
-f <path>/<file_name> ①

```

- ① Specify the path and file name, such as `~/.ssh/id_rsa`, of the new SSH key. If you have an existing key pair, ensure your public key is in the your `~/.ssh` directory.

Running this command generates an SSH key that does not require a password in the location that you specified.

**NOTE**

If you plan to install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries on the **x86_64** architecture, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"
```

Example output

```
Agent pid 31874
```

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

1. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> 1
```

Example output

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

- 1** Specify the path and file name for your SSH private key, such as `~/.ssh/id_rsa`

2. Set the **GOOGLE_APPLICATION_CREDENTIALS** environment variable to the full path to your service account private key file.

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```
$ gcloud auth list
```

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide this key to your cluster's machines.

12.7.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

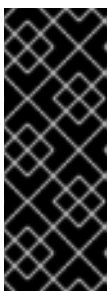
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.
- Obtain the **imageContentSources** section from the output of the command to mirror the repository.
- Obtain the contents of the certificate for your mirror registry.

Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



IMPORTANT

You must create a directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

- Unless you use a registry that RHCOS trusts by default, such as **docker.io**, you must provide the contents of the certificate for your mirror repository in the **additionalTrustBundle** section. In most cases, you must provide the certificate for your mirror.
 - You must include the **imageContentSources** section from the output of the command to mirror the repository.
3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

12.7.8.1. Sample **install-config.yaml** file for VMware vSphere

You can customize the **install-config.yaml** file to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

```
apiVersion: v1
baseDomain: example.com 1
compute:
- hyperthreading: Enabled 2 3
```


perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.

- 7 The number of control plane machines that you add to the cluster. Because the cluster uses this values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- 8 The cluster name that you specified in your DNS records.
- 9 The fully-qualified host name or IP address of the vCenter server.
- 10 The name of the user for accessing the server. This user must have at least the roles and privileges that are required for [static or dynamic persistent volume provisioning](#) in vSphere.
- 11 The password associated with the vSphere user.
- 12 The vSphere datacenter.
- 13 The default vSphere datastore to use.
- 14 Optional: For installer-provisioned infrastructure, the absolute path of an existing folder where the installation program creates the virtual machines, for example, `/<datacenter_name>/vm/<folder_name>/<subfolder_name>`. If you do not provide this value, the installation program creates a top-level folder in the datacenter virtual machine folder that is named with the infrastructure ID. If you are providing the infrastructure for the cluster, omit this parameter.
- 15 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.
- 16 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

- 17 Provide the contents of the certificate file that you used for your mirror registry.
- 18 Provide the **imageContentSources** section from the output of the command to mirror the repository.

12.7.8.2. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

Prerequisites

- An existing **install-config.yaml** file.

- Review the sites that your cluster requires access to and determine whether any need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. Add sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

- Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
  ...
```

- A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.
- A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.
- A comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass proxy for all destinations.
- If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

**NOTE**

The installation program does not support the proxy **readinessEndpoints** field.

2. Save the file and reference it when installing OpenShift Container Platform.

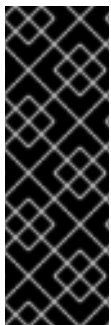
The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

**NOTE**

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

12.7.9. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

**IMPORTANT**

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

Prerequisites

- Obtain the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- Create the **install-config.yaml** installation configuration file.

Procedure

1. Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

Example output

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
```

- 1** For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

Because you create your own compute machines later in the installation process, you can safely ignore this warning.

- Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

- Optional: If you do not want the cluster to provision compute machines, remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

- Remove the Kubernetes manifest files that define the control plane machines and compute machine sets:

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

Because you create and manage these resources yourself, you do not have to initialize them.

- You can preserve the machine set files to create compute machines by using the machine API, but you must update references to them to match your environment.
- Modify the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` Kubernetes manifest file to prevent pods from being scheduled on the control plane machines:
 - Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yml` file.
 - Locate the `mastersSchedulable` parameter and set its value to **False**.
 - Save and exit the file.
 - Optional: If you do not want [the Ingress Operator](#) to create DNS records on your behalf, remove the `privateZone` and `publicZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

- 1 2 Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

7. Obtain the Ignition config files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

- 1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.7.10. Configuring chrony time service

You must set the time server and related settings used by the chrony time service (**chronyd**) by modifying the contents of the **chrony.conf** file and passing those contents to your nodes as a machine config.

Procedure

1. Create the contents of the **chrony.conf** file and encode it as base64. For example:

```
$ cat << EOF | base64
pool 0.rhel.pool.ntp.org iburst 1
driftfile /var/lib/chrony/drift
makestep 1.0 3
rtcsync
logdir /var/log/chrony
EOF
```

- 1 Specify any valid, reachable time source, such as the one provided by your DHCP server.

Example output

```
ICAgIHNIcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXlvcGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2
RpciAv
dmFyL2xvZy9jaHJvbnkK
```

2. Create the **MachineConfig** object file, replacing the base64 string with the one you just created. This example adds the file to **master** nodes. You can change it to **worker** or make an additional MachineConfig for the **worker** role. Create MachineConfig files for each type of

machine that your cluster uses:

```
$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ICAgIHNIcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxIIiC92Y
XlVbGliL2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAg
IGxvZ2RpciAvdmFyL2xvZy9jaHJvbnkK
          verification: {}
      filesystem: root
      mode: 420
      path: /etc/chrony.conf
    osImageURL: ""
EOF
```

3. Make a backup copy of the configuration files.
4. Apply the configurations in one of two ways:
 - If the cluster is not up yet, after you generate manifest files, add this file to the **<installation_directory>/openshift** directory, and then continue to create the cluster.
 - If the cluster is already running, apply the file:

```
$ oc apply -f ./99-masters-chrony-configuration.yaml
```

12.7.11. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in VMware vSphere (vSphere). The provided {cp-template} templates contain references to this infrastructure name, so you must extract it.

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in VMware vSphere. If you plan to use the cluster identifier as the name of your virtual machine folder, you must extract it.

Prerequisites

- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.
- Generate the Ignition config files for your cluster.
- Install the **jq** package.

Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

- 1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
openshift-vw9j6 1
```

- 1 The output of this command is your cluster name and a random string.

12.7.12. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

Before you install a cluster that contains user-provisioned infrastructure on VMware vSphere, you must create RHCOS machines on vSphere hosts for it to use.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that you can access from your computer and that the machines that you create can access.
- Create a [vSphere cluster](#).

Procedure

1. Upload the bootstrap Ignition config file, which is named **<installation_directory>/bootstrap.ign**, that the installation program created to your HTTP server. Note the URL of this file.
You must host the bootstrap Ignition config file because it is too large to fit in a vApp property.
2. Save the following secondary Ignition config file for your bootstrap node to your computer as **<installation_directory>/append-bootstrap.ign**.

```
{
  "ignition": {
    "config": {
      "append": [
        {
```

```

    "source": "<bootstrap_ignition_config_url>", ❶
    "verification": {}
  }
]
},
"timeouts": {},
"version": "2.2.0"
},
"networkd": {},
"passwd": {},
"storage": {},
"systemd": {}
}

```

- ❶ Specify the URL of the bootstrap Ignition config file that you hosted.

When you create the virtual machine (VM) for the bootstrap machine, you use this Ignition config file.

- Convert the master, worker, and secondary bootstrap Ignition config files to base64 encoding. For example, if you use a Linux operating system, you can use the **base64** command to encode the files.

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64
```



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

- Obtain the RHCOS OVA image. Images are available from the [RHCOS image mirror](#) page.



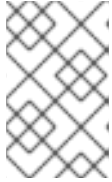
IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download an image with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image version that matches your OpenShift Container Platform version if it is available.

The filename contains the OpenShift Container Platform version number in the format **rhcos-vmware.<architecture>.ova**.

- In the vSphere Client, create a folder in your datacenter to store your VMs.
 - Click the **VMs and Templates** view.

- b. Right-click the name of your datacenter.
 - c. Click **New Folder → New VM and Template Folder**.
 - d. In the window that is displayed, enter the folder name. If you did not specify an existing folder in the **install-config.yaml** file, create a folder with the same name as the infrastructure ID.
6. In the vSphere Client, create a template for the OVA image and then clone the template as needed.



NOTE

In the following steps, you create a template and then clone the template for all of your cluster machines. You then provide the location for the Ignition config file for that cloned machine type when you provision the VMs.

- a. From the **Hosts and Clusters** tab, right-click your cluster name and select **Deploy OVF Template**.
- b. On the **Select an OVF** tab, specify the name of the RHCOS OVA file that you downloaded.
- c. On the **Select a name and folder** tab, set a **Virtual machine name** for your template, such as **Template-RHCOS**. Click the name of your vSphere cluster and select the folder you created in the previous step.
- d. On the **Select a compute resource** tab, click the name of your vSphere cluster.
- e. On the **Select storage** tab, configure the storage options for your VM.
 - Select **Thin Provision** or **Thick Provision**, based on your storage preferences.
 - Select the datastore that you specified in your **install-config.yaml** file.
- f. On the **Select network** tab, specify the network that you configured for the cluster, if available.
- g. When creating the OVF template, do not specify values on the **Customize template** tab or configure the template any further.



IMPORTANT

Do not start the original VM template. The VM template must remain off and must be cloned for new RHCOS machines. Starting the VM template configures the VM template as a VM on the platform, which prevents it from being used as a template that machine sets can apply configurations to.

7. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **control-plane-0** or **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.

- d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - Optional: In the event of cluster performance issues, from the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
 - **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
 - Alternatively, prior to powering on the virtual machine add via vApp properties:
 - Navigate to a virtual machine from the vCenter Server inventory.
 - On the **Configure** tab, expand **Settings** and select **vApp options**.
 - Scroll down and under **Properties** apply the configurations from above.
 - h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type.
 - i. Complete the configuration and power on the VM.
8. Create the rest of the machines for your cluster by following the preceding steps for each machine.



IMPORTANT

You must create the bootstrap and control plane machines at this time. Because some pods are deployed on compute machines by default, also create at least two compute machines before you install the cluster.

12.7.13. Creating more Red Hat Enterprise Linux CoreOS (RHCOS) machines in vSphere

You can create more compute machines for your cluster that uses user-provisioned infrastructure on VMware vSphere.

Prerequisites

- Obtain the base64-encoded Ignition file for your compute machines.
- You have access to the vSphere template that you created for your cluster.

Procedure

1. After the template deploys, deploy a VM for a machine in the cluster.
 - a. Right-click the template's name and click **Clone → Clone to Virtual Machine**
 - b. On the **Select a name and folder** tab, specify a name for the VM. You might include the machine type in the name, such as **compute-1**.
 - c. On the **Select a name and folder** tab, select the name of the folder that you created for the cluster.
 - d. On the **Select a compute resource** tab, select the name of a host in your datacenter.
 - e. Optional: On the **Select storage** tab, customize the storage options.
 - f. On the **Select clone options**, select **Customize this virtual machine's hardware**
 - g. On the **Customize hardware** tab, click **VM Options → Advanced**.
 - From the **Latency Sensitivity** list, select **High**.
 - Click **Edit Configuration**, and on the **Configuration Parameters** window, click **Add Configuration Params**. Define the following parameter names and values:
 - **guestinfo.ignition.config.data**: Paste the contents of the base64-encoded compute Ignition config file for this machine type.
 - **guestinfo.ignition.config.data.encoding**: Specify **base64**.
 - **disk.EnableUUID**: Specify **TRUE**.
 - h. In the **Virtual Hardware** panel of the **Customize hardware** tab, modify the specified values as required. Ensure that the amount of RAM, CPU, and disk storage meets the minimum requirements for the machine type. Also, make sure to select the correct network under **Add network adapter** if there are multiple networks available.
 - i. Complete the configuration and power on the VM.
2. Continue to create more compute machines for your cluster.

12.7.14. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ ❶
--log-level=info ❷
```

❶ For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

❷ To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After bootstrap process is complete, remove the bootstrap machine from the load balancer.



IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the machine itself.

12.7.15. Logging in to the cluster

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig ❶
```

❶ For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

12.7.16. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE   VERSION
master-0  Ready   master   63m   v1.18.3
master-1  Ready   master   63m   v1.18.3
master-2  Ready   master   64m   v1.18.3
worker-0  NotReady worker   76s   v1.18.3
worker-1  NotReady worker   70s   v1.18.3
```

The output lists all of the machines that you created.

- Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-8b2br  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
csr-8vnps  15m   system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

- If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:



NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. Once the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

- Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME      AGE   REQUESTOR                                     CONDITION
csr-bfd72 5m26s system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv 5m26s system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

- If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

- 1** **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

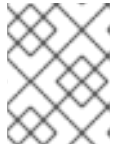
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{"\n"}{{end}}{{end}}' | xargs oc adm certificate approve
```

- After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES    AGE   VERSION
master-0  Ready    master   73m   v1.20.0
master-1  Ready    master   73m   v1.20.0
master-2  Ready    master   74m   v1.20.0
worker-0  Ready    worker   11m   v1.20.0
worker-1  Ready    worker   11m   v1.20.0
```



NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see [Certificate Signing Requests](#).

12.7.17. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME              VERSION AVAILABLE PROGRESSING DEGRADED
SINCE
authentication    4.5.4 True      False      False      69s
cloud-credential  4.5.4 True      False      False      12m
cluster-autoscaler 4.5.4 True      False      False      11m
console           4.5.4 True      False      False      46s
dns               4.5.4 True      False      False      11m
image-registry    4.5.4 True      False      False      5m26s
ingress           4.5.4 True      False      False      5m36s
kube-apiserver    4.5.4 True      False      False      8m53s
kube-controller-manager 4.5.4 True      False      False      7m24s
kube-scheduler    4.5.4 True      False      False      12m
machine-api       4.5.4 True      False      False      12m
machine-config    4.5.4 True      False      False      7m36s
marketplace       4.5.4 True      False      False      7m54m
monitoring        4.5.4 True      False      False      7h54s
```

network	4.5.4	True	False	False	5m9s
node-tuning	4.5.4	True	False	False	11m
openshift-apiserver	4.5.4	True	False	False	11m
openshift-controller-manager	4.5.4	True	False	False	5m943s
openshift-samples	4.5.4	True	False	False	3m55s
operator-lifecycle-manager	4.5.4	True	False	False	11m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	11m
service-ca	4.5.4	True	False	False	11m
service-catalog-apiserver	4.5.4	True	False	False	5m26s
service-catalog-controller-manager	4.5.4	True	False	False	5m25s
storage	4.5.4	True	False	False	5m30s

2. Configure the Operators that are not available.

12.7.17.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

12.7.17.1.1. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.



IMPORTANT

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.



NOTE

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim: 1
```

- 1** Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

12.7.17.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec": {"storage":{"emptyDir":{}}}'
```

**WARNING**

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

12.7.17.1.3. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.

**IMPORTANT**

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec": {"rolloutStrategy": "Recreate", "replicas": 1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.
 - a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage ❶
spec:
  accessModes:
  - ReadWriteOnce ❷
resources:
  requests:
    storage: 100Gi ❸
```

- ❶ A unique name that represents the **PersistentVolumeClaim** object.
- ❷ The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.
- ❸ The size of the PersistentVolumeClaim.

- b. Create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:
  pvc:
    claim: ❶
```

- ❶ Creating a custom PVC allows you to leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see [Configuring the registry for vSphere](#).

12.7.18. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.4	True	False	False	7m56s
cloud-credential	4.5.4	True	False	False	31m
cluster-autoscaler	4.5.4	True	False	False	16m
console	4.5.4	True	False	False	10m
csi-snapshot-controller	4.5.4	True	False	False	16m
dns	4.5.4	True	False	False	22m
etcd	4.5.4	False	False	False	25s
image-registry	4.5.4	True	False	False	16m
ingress	4.5.4	True	False	False	16m
insights	4.5.4	True	False	False	17m
kube-apiserver	4.5.4	True	False	False	19m
kube-controller-manager	4.5.4	True	False	False	20m
kube-scheduler	4.5.4	True	False	False	20m
kube-storage-version-migrator	4.5.4	True	False	False	16m
machine-api	4.5.4	True	False	False	22m
machine-config	4.5.4	True	False	False	22m
marketplace	4.5.4	True	False	False	16m
monitoring	4.5.4	True	False	False	10m
network	4.5.4	True	False	False	23m
node-tuning	4.5.4	True	False	False	23m
openshift-apiserver	4.5.4	True	False	False	17m
openshift-controller-manager	4.5.4	True	False	False	15m
openshift-samples	4.5.4	True	False	False	16m
operator-lifecycle-manager	4.5.4	True	False	False	22m
operator-lifecycle-manager-catalog	4.5.4	True	False	False	22m
operator-lifecycle-manager-packageserver	4.5.4	True	False	False	18m
service-ca	4.5.4	True	False	False	23m
service-catalog-apiserver	4.5.4	True	False	False	23m
service-catalog-controller-manager	4.5.4	True	False	False	23m
storage	4.5.4	True	False	False	17m

Alternatively, the following command notifies you when all of the clusters are available. It also retrieves and displays credentials:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

Example output

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.



IMPORTANT

The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

2. Confirm that the Kubernetes API server is communicating with the pods.

- a. To view a list of all pods, use the following command:

```
$ oc get pods --all-namespaces
```

Example output

```

NAMESPACE                NAME                                READY  STATUS
RESTARTS  AGE
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1
Running   1    9m
openshift-apiserver          apiserver-67b9g                                1/1  Running  0
3m
openshift-apiserver          apiserver-ljcmx                                1/1  Running  0
1m
openshift-apiserver          apiserver-z25h4                                1/1  Running  0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8  1/1
Running   0    5m
...

```

- b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace> 1
```

- 1** Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

3. Register your cluster on the [Cluster registration](#) page.

You can add extra compute machines after the cluster installation is completed by following [Adding compute machines to vSphere](#).

12.7.19. Backing up VMware vSphere volumes

OpenShift Container Platform provisions new volumes as independent persistent disks to freely attach and detach the volume on any node in the cluster. As a consequence, it is not possible to back up volumes that use snapshots, or to restore volumes from snapshots. See [Snapshot Limitations](#) for more information.

Procedure

To create a backup of persistent volumes:

1. Stop the application that is using the persistent volume.
2. Clone the persistent volume.
3. Restart the application.
4. Create a backup of the cloned volume.
5. Delete the cloned volume.

12.7.20. Next steps

- [Customize your cluster](#).
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by [configuring additional trust stores](#).
- If necessary, you can [opt out of remote health reporting](#).

12.8. UNINSTALLING A CLUSTER ON VSPHERE THAT USES INSTALLER-PROVISIONED INFRASTRUCTURE

You can remove a cluster that you deployed in your VMware vSphere instance by using installer-provisioned infrastructure.

12.8.1. Removing a cluster that uses installer-provisioned infrastructure

You can remove a cluster that uses installer-provisioned infrastructure from your cloud.



NOTE

After uninstallation, check your cloud provider for any resources not removed properly, especially with User Provisioned Infrastructure (UPI) clusters. There might be resources that the installer did not create or that the installer is unable to access. For example, some Google Cloud resources require [IAM permissions](#) in shared VPC host projects, or there might be unused [health checks that must be deleted](#).

Prerequisites

- Have a copy of the installation program that you used to deploy the cluster.

- Have the files that the installation program generated when you created your cluster.

Procedure

1. From the computer that you used to install the cluster, run the following command:

```
$. /openshift-install destroy cluster \  
--dir=<installation_directory> --log-level=info 1 2
```

- 1** For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
- 2** To view different details, specify **warn**, **debug**, or **error** instead of **info**.



NOTE

You must specify the directory that contains the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

CHAPTER 13. INSTALLATION CONFIGURATION

13.1. SUPPORTED INSTALLATION METHODS FOR DIFFERENT PLATFORMS

You can perform different types of installations on different platforms.



NOTE

Not all installation options are supported for all platforms, as shown in the following tables.

Table 13.1. Installer-provisioned infrastructure options

	AWS	Azure	GCP	OpenStack	RHV	Bare metal	vSphere	IBM Z
Default	X	X	X		X		X	
Custom	X	X	X	X	X		X	
Network Operator	X	X	X				X	
Restricted network	X		X	X			X	
Private clusters	X	X	X					
Existing virtual private networks	X	X	X					

Table 13.2. User-provisioned infrastructure options

	AWS	Azure	GCP	OpenStack	RHV	Bare metal	vSphere	IBM Z
Custom	X	X	X	X		X	X	
Network Operator						X	X	

	AWS	Azure	GCP	OpenSt ack	RHV	Bare metal	vSphere	IBM Z
Restrict ed network	X		X			X	X	
Shared VPC hosted outside of cluster project			X					

13.2. CUSTOMIZING NODES

Although directly making changes to OpenShift Container Platform nodes is discouraged, there are times when it is necessary to implement a required low-level security, networking, or performance feature. Direct changes to OpenShift Container Platform nodes can be done by:

- Creating machine configs that are included in manifest files to start up a cluster during **openshift-install**.
- Creating machine configs that are passed to running OpenShift Container Platform nodes via the Machine Config Operator.

The following sections describe features that you might want to configure on your nodes in this way.

13.2.1. Adding day-1 kernel arguments

Although it is often preferable to modify kernel arguments as a day-2 activity, you might want to add kernel arguments to all master or worker nodes during initial cluster installation. Here are some reasons you might want to add kernel arguments during cluster installation so they take effect before the systems first boot up:

- You want to disable a feature, such as SELinux, so it has no impact on the systems when they first come up.
- You need to do some low-level network configuration before the systems start.

To add kernel arguments to master or worker nodes, you can create a **MachineConfig** object and inject that object into the set of manifest files used by Ignition during cluster setup.

For a listing of arguments you can pass to a RHEL 8 kernel at boot time, see [Kernel.org kernel parameters](#). It is best to only add kernel arguments with this procedure if they are needed to complete the initial OpenShift Container Platform installation.

Procedure

1. Generate the Kubernetes manifests for the cluster:


```
$ ./openshift-install create manifests --dir=<installation_directory>
```

2. Decide if you want to add kernel arguments to worker or master nodes.
3. In the **openshift** directory, create a file (for example, **99-openshift-machineconfig-master-kargs.yaml**) to define a **MachineConfig** object to add the kernel settings. This example adds a **loglevel=7** kernel argument to master nodes:

```
$ cat << EOF > 99-openshift-machineconfig-master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - 'loglevel=7'
EOF
```

You can change **master** to **worker** to add kernel arguments to worker nodes instead. Create a separate YAML file to add to both master and worker nodes.

You can now continue on to create the cluster.

13.2.2. Adding kernel modules to nodes

For most common hardware, the Linux kernel includes the device driver modules needed to use that hardware when the computer starts up. For some hardware, however, modules are not available in Linux. Therefore, you must find a way to provide those modules to each host computer. This procedure describes how to do that for nodes in an OpenShift Container Platform cluster.

When a kernel module is first deployed by following these instructions, the module is made available for the current kernel. If a new kernel is installed, the `kmods-via-containers` software will rebuild and deploy the module so a compatible version of that module is available with the new kernel.

The way that this feature is able to keep the module up to date on each node is by:

- Adding a systemd service to each node that starts at boot time to detect if a new kernel has been installed and
- If a new kernel is detected, the service rebuilds the module and installs it to the kernel

For information on the software needed for this procedure, see the [kmods-via-containers](#) github site.

A few important issues to keep in mind:

- This procedure is Technology Preview.
- Software tools and examples are not yet available in official RPM form and can only be obtained for now from unofficial **github.com** sites noted in the procedure.
- Third-party kernel modules you might add through these procedures are not supported by Red Hat.
- In this procedure, the software needed to build your kernel modules is deployed in a RHEL 8

container. Keep in mind that modules are rebuilt automatically on each node when that node gets a new kernel. For that reason, each node needs access to a **yum** repository that contains the kernel and related packages needed to rebuild the module. That content is best provided with a valid RHEL subscription.

13.2.2.1. Building and testing the kernel module container

Before deploying kernel modules to your OpenShift Container Platform cluster, you can test the process on a separate RHEL system. Gather the kernel module's source code, the KVC framework, and the `kmod-via-containers` software. Then build and test the module. To do that on a RHEL 8 system, do the following:

Procedure

1. Register a RHEL 8 system:

```
# subscription-manager register
```

2. Attach a subscription to the RHEL 8 system:

```
# subscription-manager attach --auto
```

3. Install software that is required to build the software and container:

```
# yum install podman make git -y
```

4. Clone the **kmod-via-containers** repository:

- a. Create a folder for the repository:

```
$ mkdir kmods; cd kmods
```

- b. Clone the repository:

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

5. Install a KVC framework instance on your RHEL 8 build host to test the module. This adds a **kmods-via-container** systemd service and loads it:

- a. Change to the **kmod-via-containers** directory:

```
$ cd kmods-via-containers/
```

- b. Install the KVC framework instance:

```
$ sudo make install
```

- c. Reload the systemd manager configuration:

```
$ sudo systemctl daemon-reload
```

6. Get the kernel module source code. The source code might be used to build a third-party module that you do not have control over, but is supplied by others. You will need content

similar to the content shown in the **kvc-simple-kmod** example that can be cloned to your system as follows:

```
$ cd .. ; git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

7. Edit the configuration file, **simple-kmod.conf** file, in this example, and change the name of the Dockerfile to **Dockerfile.rhel**:

- a. Change to the **kvc-simple-kmod** directory:

```
$ cd kvc-simple-kmod
```

- b. Rename the Dockerfile:

```
$ cat simple-kmod.conf
```

Example Dockerfile

```
KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-
simple-kmod.git"
KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

8. Create an instance of **kmods-via-containers@.service** for your kernel module, **simple-kmod** in this example:

```
$ sudo make install
```

9. Enable the **kmods-via-containers@.service** instance:

```
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

10. Enable and start the systemd service:

- a. Enable the service:

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service
```

- b. Start the service:

```
$ sudo systemctl start kmods-via-containers@simple-kmod.service
```

- c. Review the service status:

```
$ sudo systemctl status kmods-via-containers@simple-kmod.service
```

Example output

- kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
enabled; vendor preset: disabled)

```
Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

- To confirm that the kernel modules are loaded, use the **lsmod** command to list the modules:

```
$ lsmod | grep simple_
```

Example output

```
simple_procfs_kmod 16384 0
simple_kmod 16384 0
```

- Optional. Use other methods to check that the **simple-kmod** example is working:

- Look for a "Hello world" message in the kernel ring buffer with **dmesg**:

```
$ dmesg | grep 'Hello world'
```

Example output

```
[ 6420.761332] Hello world from simple_kmod.
```

- Check the value of **simple-procfs-kmod** in **/proc**:

```
$ sudo cat /proc/simple-procfs-kmod
```

Example output

```
simple-procfs-kmod number = 0
```

- Run the **spkut** command to get more information from the module:

```
$ sudo spkut 44
```

Example output

```
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
+ podman run -i --rm --privileged
  simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

Going forward, when the system boots this service will check if a new kernel is running. If there is a new kernel, the service builds a new version of the kernel module and then loads it. If the module is already built, it will just load it.

13.2.2.2. Provisioning a kernel module to OpenShift Container Platform

Depending on whether or not you must have the kernel module in place when OpenShift Container Platform cluster first boots, you can set up the kernel modules to be deployed in one of two ways:

- **Provision kernel modules at cluster install time (day-1)** You can create the content as a **MachineConfig** object and provide it to **openshift-install** by including it with a set of manifest files.
- **Provision kernel modules via Machine Config Operator (day-2)** If you can wait until the cluster is up and running to add your kernel module, you can deploy the kernel module software via the Machine Config Operator (MCO).

In either case, each node needs to be able to get the kernel packages and related software packages at the time that a new kernel is detected. There are a few ways you can set up each node to be able to obtain that content.

- Provide RHEL entitlements to each node.
- Get RHEL entitlements from an existing RHEL host, from the **/etc/pki/entitlement** directory and copy them to the same location as the other files you provide when you build your Ignition config.
- Inside the Dockerfile, add pointers to a **yum** repository containing the kernel and other packages. This must include new kernel packages as they are needed to match newly installed kernels.

13.2.2.2.1. Provision kernel modules via a **MachineConfig** object

By packaging kernel module software with a **MachineConfig** object, you can deliver that software to worker or master nodes at installation time or via the Machine Config Operator.

First create a base Ignition config that you would like to use. At installation time, the Ignition config will contain the ssh public key to add to the **authorized_keys** file for the **core** user on the cluster. To add the **MachineConfig** object later via the MCO instead, the SSH public key is not required. For both type, the example simple-kmod service creates a systemd unit file, which requires a **kmods-via-containers@simple-kmod.service**.



NOTE

The systemd unit is a workaround for an [upstream bug](#) and makes sure that the **kmods-via-containers@simple-kmod.service** gets started on boot:

1. Register a RHEL 8 system:

```
# subscription-manager register
```

2. Attach a subscription to the RHEL 8 system:

```
# subscription-manager attach --auto
```

3. Install software needed to build the software:

```
# yum install podman make git -y
```

4. Create an Ignition config file that creates a systemd unit file:

- a. Create a directory to host the Ignition config file:

```
$ mkdir kmods; cd kmods
```

-
- b. Create the Ignition config file that creates a systemd unit file:

```
$ cat <<EOF > ./baseconfig.ign
{
  "ignition": { "version": "2.2.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "groups": ["sudo"],
        "sshAuthorizedKeys": [
          "ssh-rsa AAAA"
        ]
      }
    ]
  },
  "systemd": {
    "units": [{
      "name": "require-kvc-simple-kmod.service",
      "enabled": true,
      "contents": "[Unit]\nRequires=kmods-via-containers@simple-
kmod.service\n[Service]\nType=oneshot\nExecStart=/usr/bin/true\n\n[Install]\nWantedBy=m
ulti-user.target"
    }]
  }
}
EOF
```



NOTE

You must add your public SSH key to the **baseconfig.ign** file to use the file during **openshift-install**. The public SSH key is not needed if you create the **MachineConfig** object using the MCO.

5. Create a base MCO YAML snippet that uses the following configuration:

```
$ cat <<EOF > mc-base.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 10-kvc-simple-kmod
spec:
  config:
EOF
```



NOTE

The **mc-base.yaml** is set to deploy the kernel module on **worker** nodes. To deploy on master nodes, change the role from **worker** to **master**. To do both, you could repeat the whole procedure using different file names for the two types of deployments.

6. Get the **kmods-via-containers** software:

a. Clone the **kmods-via-containers** repository:

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

b. Clone the **kvc-simple-kmod** repository:

```
$ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

7. Get your module software. In this example, **kvc-simple-kmod** is used:

8. Create a fakeroot directory and populate it with files that you want to deliver via Ignition, using the repositories cloned earlier:

a. Create the directory:

```
$ FAKEROOT=$(mktemp -d)
```

b. Change to the **kmod-via-containers** directory:

```
$ cd kmods-via-containers
```

c. Install the KVC framework instance:

```
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

d. Change to the **kvc-simple-kmod** directory:

```
$ cd ../kvc-simple-kmod
```

e. Create the instance:

```
$ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
```

9. Get a tool called **filetranspiler** and dependent software:

```
$ cd .. ; sudo yum install -y python3
git clone https://github.com/ashcrow/filetranspiler.git
```

10. Generate a final machine config YAML (**mc.yaml**) and have it include the base Ignition config, base machine config, and the fakeroot directory with files you would like to deliver:

```
$ ./filetranspiler/filetranspile -i ./baseconfig.ign \
  -f ${FAKEROOT} --format=yaml --dereference-symlinks \
  | sed 's/^ / /' | (cat mc-base.yaml -) > 99-simple-kmod.yaml
```

11. If the cluster is not up yet, generate manifest files and add this file to the **openshift** directory. If the cluster is already running, apply the file as follows:

```
$ oc create -f 99-simple-kmod.yaml
```

Your nodes will start the **kmods-via-containers@simple-kmod.service** service and the kernel modules will be loaded.

- To confirm that the kernel modules are loaded, you can log in to a node (using **oc debug node/<openshift-node>**, then **chroot /host**). To list the modules, use the **lsmod** command:

```
$ lsmod | grep simple_
```

Example output

```
simple_procfs_kmod 16384 0
simple_kmod        16384 0
```

13.2.3. Encrypting disks during installation

During OpenShift Container Platform installation, you can enable disk encryption on all master and worker nodes. This feature:

- Is available for installer-provisioned infrastructure and user-provisioned infrastructure deployments
- Is supported on Red Hat Enterprise Linux CoreOS (RHCOS) systems only
- Sets up disk encryption during the manifest installation phase so all data written to disk, from first boot forward, is encrypted
- Encrypts data on the root filesystem only (**/dev/mapper/coreos-luks-root** on */*)
- Requires no user intervention for providing passphrases
- Uses AES-256-CBC encryption

There are two different supported encryption modes:

- TPM v2:** This is the preferred mode. TPM v2 stores passphrases in a secure cryptoprocessor. To implement TPM v2 disk encryption, create an Ignition config file as described below.
- Tang:** To use Tang to encrypt your cluster, you need to use a Tang server. Clevis implements decryption on the client side. Tang encryption mode is only supported for bare metal installs.

Follow one of the two procedures to enable disk encryption for the nodes in your cluster.

13.2.3.1. Enabling TPM v2 disk encryption

Use this procedure to enable TPM v2 mode disk encryption during OpenShift Container Platform deployment.

Procedure

- Check to see if TPM v2 encryption needs to be enabled in the BIOS on each node. This is required on most Dell systems. Check the manual for your computer.
- Generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory>
```


3. In the **openshift** directory, create master or worker files to encrypt disks for those nodes.

- To create a worker file, run the following command:

```
$ cat << EOF > ./99-openshift-worker-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tpm
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

- To create a master file, run the following command:

```
$ cat << EOF > ./99-openshift-master-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tpm
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

4. Make a backup copy of the YAML file. You should do this because the file will be deleted when you create the cluster.
5. Continue with the remainder of the OpenShift Container Platform deployment.

13.2.3.2. Enabling Tang disk encryption

Use this procedure to enable Tang mode disk encryption during OpenShift Container Platform deployment.

Procedure

1. Access a Red Hat Enterprise Linux server from which you can configure the encryption settings and run **openshift-install** to install a cluster and **oc** to work with it.
2. Set up or access an existing Tang server. See [Network-bound disk encryption](#) for instructions. See [Securing Automated Decryption New Cryptography and Techniques](#) for a presentation on Tang.
3. Add kernel arguments to configure networking when you do the Red Hat Enterprise Linux CoreOS (RHCOS) installations for your cluster. For example, to configure DHCP networking, identify **ip=dhcp**, or set static networking when you add parameters to the kernel command line. For both DHCP and static networking, you also must provide the **rd.neednet=1** kernel argument.



IMPORTANT

Skipping this step causes the second boot to fail.

4. Install the **clevis** package, if it is not already installed:

```
$ sudo yum install clevis -y
```

5. Generate a thumbprint from the Tang server.

- a. In the following command, replace the value of **url** with the Tang server URL:

```
$ echo nifty random wordwords \  
  | clevis-encrypt-tang \  
  '{"url":"https://tang.example.org"}'
```

Example output

The advertisement contains the following signing keys:

```
PLjNyRdGw03zIRoGjQYMahSZGu9
```

- b. When the **Do you wish to trust these keys? [ynYN]** prompt displays, type **Y**, and the thumbprint is displayed:

Example output

```
eyJhbmc3SIRyMXpPenc3ajhEQ01tZVJiT11oM...
```

6. Create a Base64 encoded file, replacing the URL of the Tang server (**url**) and thumbprint (**thp**) you just generated:

```
$ (cat <<EOM  
{  
  "url": "https://tang.example.com",  
  "thp": "PLjNyRdGw03zIRoGjQYMahSZGu9"  
}  
EOM  
) | base64 -w0
```

Example output

```
ewoglnVybCl6lCJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAoglnRocCl6lCJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAlCn0K
```

7. In the **openshift** directory, create master or worker files to encrypt disks for those nodes.

- For worker nodes, use the following command:

```
$ cat << EOF > ./99-openshift-worker-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tang
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        source:
          data:text/plain;base64,ewoglnVybCl6lCJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAoglnRocCl6lCJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAlCn0K
      filesystem: root
      mode: 420
      path: /etc/clevis.json
EOF
```

- For master nodes, use the following command:

```
$ cat << EOF > ./99-openshift-master-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tang
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        source:
          data:text/plain;base64,ewoglnVybCl6lCJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAoglnRocCl6lCJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAlCn0K
      filesystem: root
```


2. Create the **MachineConfig** object file, replacing the base64 string with the one you just created. This example adds the file to **master** nodes. You can change it to **worker** or make an additional MachineConfig for the **worker** role. Create MachineConfig files for each type of machine that your cluster uses:

```
$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ICAgIHNIcnZlciBjbG9jay5yZWROYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxIC92Y
XlVbGliL2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAg
IGxvZ2RpciAvdmFyL2xvZy9jaHJvbnkK
          verification: {}
          filesystem: root
          mode: 420
          path: /etc/chrony.conf
      osImageURL: ""
EOF
```

3. Make a backup copy of the configuration files.
4. Apply the configurations in one of two ways:
 - If the cluster is not up yet, after you generate manifest files, add this file to the **<installation_directory>/openshift** directory, and then continue to create the cluster.
 - If the cluster is already running, apply the file:

```
$ oc apply -f ./99-masters-chrony-configuration.yaml
```

13.2.5. Additional resources

See [Support for FIPS cryptography](#) for information on FIPS support.

13.3. AVAILABLE CLUSTER CUSTOMIZATIONS

You complete most of the cluster configuration and customization after you deploy your OpenShift Container Platform cluster. A number of *configuration resources* are available.

You modify the configuration resources to configure the major features of the cluster, such as the image registry, networking configuration, image build behavior, and the identity provider.

For current documentation of the settings that you control by using these resources, use the **oc explain** command, for example **oc explain builds --api-version=config.openshift.io/v1**

13.3.1. Cluster configuration resources

All cluster configuration resources are globally scoped (not namespaced) and named **cluster**.

Resource name	Description
apiserver.config.openshift.io	Provides API server configuration such as certificates and certificate authorities .
authentication.config.openshift.io	Controls the identity provider and authentication configuration for the cluster.
build.config.openshift.io	Controls default and enforced configuration for all builds on the cluster.
console.config.openshift.io	Configures the behavior of the web console interface, including the logout behavior .
featuregate.config.openshift.io	Enables FeatureGates so that you can use Tech Preview features.
image.config.openshift.io	Configures how specific image registries should be treated (allowed, disallowed, insecure, CA details).
ingress.config.openshift.io	Configuration details related to routing such as the default domain for routes.
oauth.config.openshift.io	Configures identity providers and other behavior related to internal OAuth server flows.
project.config.openshift.io	Configures how projects are created including the project template.
proxy.config.openshift.io	Defines proxies to be used by components needing external network access. Note: not all components currently consume this value.
scheduler.config.openshift.io	Configures scheduler behavior such as policies and default node selectors.

13.3.2. Operator configuration resources

These configuration resources are cluster-scoped instances, named **cluster**, which control the behavior of a specific component as owned by a particular Operator.

Resource name	Description
console.operator.openshift.io	Controls console appearance such as branding customizations
config.imageregistry.operator.openshift.io	Configures internal image registry settings such as public routing, log levels, proxy settings, resource constraints, replica counts, and storage type.
config.samples.operator.openshift.io	Configures the Samples Operator to control which example image streams and templates are installed on the cluster.

13.3.3. Additional configuration resources

These configuration resources represent a single instance of a particular component. In some cases, you can request multiple instances by creating multiple instances of the resource. In other cases, the Operator can use only a specific resource instance name in a specific namespace. Reference the component-specific documentation for details on how and when you can create additional resource instances.

Resource name	Instance name	Namespace	Description
alertmanager.monitoring.coreos.com	main	openshift-monitoring	Controls the Alertmanager deployment parameters.
ingresscontroller.operator.openshift.io	default	openshift-ingress-operator	Configures Ingress Operator behavior such as domain, number of replicas, certificates, and controller placement.

13.3.4. Informational Resources

You use these resources to retrieve information about the cluster. Do not edit these resources directly.

Resource name	Instance name	Description
clusterversion.config.openshift.io	version	In OpenShift Container Platform 4.5, you must not customize the ClusterVersion resource for production clusters. Instead, follow the process to update a cluster .
dns.config.openshift.io	cluster	You cannot modify the DNS settings for your cluster. You can view the DNS Operator status .

Resource name	Instance name	Description
infrastructure.config.openshift.io	cluster	Configuration details allowing the cluster to interact with its cloud provider.
network.config.openshift.io	cluster	You cannot modify your cluster networking after installation. To customize your network, follow the process to customize networking during installation .

13.3.5. Updating the global cluster pull secret

You can update the global pull secret for your cluster.



WARNING

Cluster resources must adjust to the new pull secret, which can temporarily limit the usability of the cluster.



WARNING

Updating the global pull secret will cause node reboots while the Machine Config Operator (MCO) syncs the changes.

Prerequisites

- You have a new or modified pull secret file to upload.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

- Enter the following command to update the global pull secret for your cluster:

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=<pull-secret-location> 1
```

- 1** Provide the path to the new pull secret file.

This update is rolled out to all nodes, which can take some time depending on the size of your cluster. During this time, nodes are drained and pods are rescheduled on the remaining nodes.

13.4. CONFIGURING YOUR FIREWALL

If you use a firewall, you must configure it so that OpenShift Container Platform can access the sites that it requires to function. You must always grant access to some sites, and you grant access to more if you use Red Hat Insights, the Telemetry service, a cloud to host your cluster, and certain build strategies.

13.4.1. Configuring your firewall for OpenShift Container Platform

Before you install OpenShift Container Platform, you must configure your firewall to grant access to the sites that OpenShift Container Platform requires.

There are no special configuration considerations for services running on only controller nodes versus worker nodes.

Procedure

1. Allowlist the following registry URLs:

URL	Port	Function
registry.redhat.io	443, 80	Provides core container images
quay.io	443, 80	Provides core container images
*.quay.io	443, 80	Provides core container images
sso.redhat.com	443, 80	The https://cloud.redhat.com/openshift site uses authentication from sso.redhat.com
openshift.org	443, 80	Provides Red Hat Enterprise Linux CoreOS (RHCOS) images

When you add a site, such as **quay.io**, to your allowlist, do not add a wildcard entry, such as ***.quay.io**, to your denylist. In most cases, image registries use a content delivery network (CDN) to serve images. If a firewall blocks access, then image downloads are denied when the initial download request is redirected to a host name such as **cdn01.quay.io**.

CDN host names, such as **cdn01.quay.io**, are covered when you add a wildcard entry, such as ***.quay.io**, in your allowlist.

2. Allowlist any site that provides resources for a language or framework that your builds require.
3. If you do not disable Telemetry, you must grant access to the following URLs to access Red Hat Insights:

URL	Port	Function
cert-api.access.redhat.com	443, 80	Required for Telemetry
api.access.redhat.com	443, 80	Required for Telemetry
infogw.api.openshift.com	443, 80	Required for Telemetry
https://cloud.redhat.com/api/ingress	443, 80	Required for Telemetry and for insights-operator

4. If you use Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) to host your cluster, you must grant access to the URLs that provide the cloud provider API and DNS for that cloud:

Cloud	URL	Port	Function
AWS	*.amazonaws.com	443, 80	Required to access AWS services and resources. Review the AWS Service Endpoints in the AWS documentation to determine the exact endpoints to allow for the regions that you use.
	oso-rhc4tp-docker-registry.s3-us-west-2.amazonaws.com	443, 80	Required to access AWS services and resources when using strict security requirements. Review the AWS Service Endpoints in the AWS documentation to determine the exact endpoints to allow for the regions that you use.
GCP	*.googleapis.com	443, 80	Required to access GCP services and resources. Review Cloud Endpoints in the GCP documentation to determine the endpoints to allow for your APIs.
	accounts.google.com	443, 80	Required to access your GCP account.
Azure	management.azure.com	443, 80	Required to access Azure services and resources. Review the Azure REST API Reference in the Azure documentation to determine the endpoints to allow for your APIs.

5. Allowlist the following URLs:

URL	Port	Function
mirror.openshift.com	443, 80	Required to access mirrored installation content and images. This site is also a source of release image signatures, although the Cluster Version Operator needs only a single functioning source.
storage.googleapis.com/openshift-release	443, 80	A source of release image signatures, although the Cluster Version Operator needs only a single functioning source.
*.apps.<cluster_name>.<base_domain>	443, 80	Required to access the default cluster routes unless you set an ingress wildcard during installation.
quay-registry.s3.amazonaws.com	443, 80	Required to access Quay image content in AWS.
api.openshift.com	443, 80	Required to check if updates are available for the cluster.
art-rhcos-ci.s3.amazonaws.com	443, 80	Required to download Red Hat Enterprise Linux CoreOS (RHCOS) images.
api.openshift.com	443, 80	Required for your cluster token.
cloud.redhat.com/openshift	443, 80	Required for your cluster token.
registry.access.redhat.com	443, 80	Required for odo CLI.

Operators require route access to perform health checks. Specifically, the authentication and web console Operators connect to two routes to verify that the routes work. If you are the cluster administrator and do not want to allow ***.apps.<cluster_name>.<base_domain>**, then allow these routes:

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**, or the host name that is specified in the **spec.route.hostname** field of the **consoles.operator/cluster** object if the field is not empty.

6. If you use a default Red Hat Network Time Protocol (NTP) server allow the following URLs:

- **1.rhel.pool.ntp.org**
- **2.rhel.pool.ntp.org**
- **3.rhel.pool.ntp.org**

**NOTE**

If you do not use a default Red Hat NTP server, verify the NTP server for your platform and allow it in your firewall.

13.5. CONFIGURING A PRIVATE CLUSTER

After you install an OpenShift Container Platform version 4.5 cluster, you can set some of its core components to be private.

**IMPORTANT**

You can configure this change for only clusters that use infrastructure that you provision to a cloud provider.

13.5.1. About private clusters

By default, OpenShift Container Platform is provisioned using publicly-accessible DNS and endpoints. You can set the DNS, Ingress Controller, and API server to private after you deploy your cluster.

DNS

If you install OpenShift Container Platform on installer-provisioned infrastructure, the installation program creates records in a pre-existing public zone and, where possible, creates a private zone for the cluster's own DNS resolution. In both the public zone and the private zone, the installation program or cluster creates DNS entries for ***.apps**, for the **Ingress** object, and **api**, for the API server.

The ***.apps** records in the public and private zone are identical, so when you delete the public zone, the private zone seamlessly provides all DNS resolution for the cluster.

Ingress Controller

Because the default **Ingress** object is created as public, the load balancer is internet-facing and in the public subnets. You can replace the default Ingress Controller with an internal one.

API server

By default, the installation program creates appropriate network load balancers for the API server to use for both internal and external traffic.

On Amazon Web Services (AWS), separate public and private load balancers are created. The load balancers are identical except that an additional port is available on the internal one for use within the cluster. Although the installation program automatically creates or destroys the load balancer based on API server requirements, the cluster does not manage or maintain them. As long as you preserve the cluster's access to the API server, you can manually modify or move the load balancers. For the public load balancer, port 6443 is open and the health check is configured for HTTPS against the **/readyz** path.

On Google Cloud Platform, a single load balancer is created to manage both internal and external API traffic, so you do not need to modify the load balancer.

On Microsoft Azure, both public and private load balancers are created. However, because of limitations in current implementation, you just retain both load balancers in a private cluster.

13.5.2. Setting DNS to private

After you deploy a cluster, you can modify its DNS to use only a private zone.

Procedure

1. Review the **DNS** custom resource for your cluster:

```
$ oc get dnses.config.openshift.io/cluster -o yaml
```

Example output

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>: owned
  publicZone:
    id: Z2XXXXXXXXXXA4
status: {}
```

Note that the **spec** section contains both a private and a public zone.

2. Patch the **DNS** custom resource to remove the public zone:

```
$ oc patch dnses.config.openshift.io/cluster --type=merge --patch='{"spec": {"publicZone": null}}'
dns.config.openshift.io/cluster patched
```

Because the Ingress Controller consults the **DNS** definition when it creates **Ingress** objects, when you create or modify **Ingress** objects, only private records are created.



IMPORTANT

DNS records for the existing Ingress objects are not modified when you remove the public zone.

3. Optional: Review the **DNS** custom resource for your cluster and confirm that the public zone was removed:

```
$ oc get dnses.config.openshift.io/cluster -o yaml
```

Example output

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
```

```

name: cluster
resourceVersion: "37966"
selfLink: /apis/config.openshift.io/v1/dnses/cluster
uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>-wfp4: owned
status: {}

```

13.5.3. Setting the Ingress Controller to private

After you deploy a cluster, you can modify its Ingress Controller to use only a private zone.

Procedure

1. Modify the default Ingress Controller to use only an internal endpoint:

```

$ oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
  loadBalancer:
    scope: Internal
EOF

```

Example output

```

ingresscontroller.operator.openshift.io "default" deleted
ingresscontroller.operator.openshift.io/default replaced

```

The public DNS entry is removed, and the private zone entry is updated.

13.5.4. Restricting the API server to private

After you deploy a cluster to Amazon Web Services (AWS) or Microsoft Azure, you can reconfigure the API server to use only the private zone.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Have access to the web console as a user with **admin** privileges.

Procedure

1. In the web portal or console for AWS or Azure, take the following actions:
 - a. Locate and delete appropriate load balancer component.
 - For AWS, delete the external load balancer. The API DNS entry in the private zone already points to the internal load balancer, which uses an identical configuration, so you do not need to modify the internal load balancer.
 - For Azure, delete the **api-internal** rule for the load balancer.
 - b. Delete the **api.\$clustername.\$yourdomain** DNS entry in the public zone.

2. From your terminal, list the cluster machines:

```
$ oc get machine -n openshift-machine-api
```

Example output

```
NAME                STATE  TYPE     REGION  ZONE     AGE
lk4pj-master-0     running m4.xlarge us-east-1 us-east-1a 17m
lk4pj-master-1     running m4.xlarge us-east-1 us-east-1b 17m
lk4pj-master-2     running m4.xlarge us-east-1 us-east-1a 17m
lk4pj-worker-us-east-1a-5fzfq running m4.xlarge us-east-1 us-east-1a 15m
lk4pj-worker-us-east-1a-vbghs running m4.xlarge us-east-1 us-east-1a 15m
lk4pj-worker-us-east-1b-zgpzg running m4.xlarge us-east-1 us-east-1b 15m
```

You modify the control plane machines, which contain **master** in the name, in the following step.

3. Remove the external load balancer from each control plane machine.
 - a. Edit a control plane **Machine** object to remove the reference to the external load balancer.

```
$ oc edit machines -n openshift-machine-api <master_name> 1
```

- 1** Specify the name of the control plane, or master, **Machine** object to modify.

- b. Remove the lines that describe the external load balancer, which are marked in the following example, and save and exit the object specification:

```
...
spec:
  providerSpec:
    value:
      ...
      loadBalancers:
        - name: lk4pj-ext 1
          type: network 2
        - name: lk4pj-int
          type: network
```

- 1** **2** Delete this line.

- c. Repeat this process for each of the machines that contains **master** in the name.

CHAPTER 14. TROUBLESHOOTING INSTALLATION ISSUES

To assist in troubleshooting a failed OpenShift Container Platform installation, you can gather logs from the bootstrap and control plane, or master, machines. You can also get debug information from the installation program.

14.1. PREREQUISITES

- You attempted to install an OpenShift Container Platform cluster, and installation failed.

14.2. GATHERING LOGS FROM A FAILED INSTALLATION

If you gave an SSH key to your installation program, you can gather data about your failed installation.



NOTE

You use a different command to gather logs about an unsuccessful installation than to gather logs from a running cluster. If you must gather logs from a running cluster, use the **oc adm must-gather** command.

Prerequisites

- Your OpenShift Container Platform installation failed before the bootstrap process finished. The bootstrap node is running and accessible through SSH.
- The **ssh-agent** process is active on your computer, and you provided the same SSH key to both the **ssh-agent** process and the installation program.
- If you tried to install a cluster on infrastructure that you provisioned, you must have the fully qualified domain names of the bootstrap and master nodes.

Procedure

- Generate the commands that are required to obtain the installation logs from the bootstrap and control plane machines:

- If you used installer-provisioned infrastructure, run the following command:

```
$ ./openshift-install gather bootstrap --dir=<installation_directory> 1
```

- 1** **installation_directory** is the directory you specified when you ran **./openshift-install create cluster**. This directory contains the OpenShift Container Platform definition files that the installation program creates.

For installer-provisioned infrastructure, the installation program stores information about the cluster, so you do not specify the host names or IP addresses.

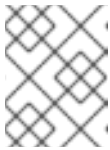
- If you used infrastructure that you provisioned yourself, run the following command:

```
$ ./openshift-install gather bootstrap --dir=<installation_directory> \ 1
  --bootstrap <bootstrap_address> \ 2
  --master <master_1_address> \ 3
```



```
--master <master_2_address> \ 4
--master <master_3_address>" 5
```

- 1 For **installation_directory**, specify the same directory you specified when you ran **./openshift-install create cluster**. This directory contains the OpenShift Container Platform definition files that the installation program creates.
- 2 **<bootstrap_address>** is the fully qualified domain name or IP address of the cluster's bootstrap machine.
- 3 4 5 For each control plane, or master, machine in your cluster, replace **<master_*_address>** with its fully qualified domain name or IP address.



NOTE

A default cluster contains three control plane machines. List all of your control plane machines as shown, no matter how many your cluster uses.

Example output

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
<timestamp>.tar.gz"
```

If you open a Red Hat support case about your installation failure, include the compressed logs in the case.

14.3. MANUALLY GATHERING LOGS WITH SSH ACCESS TO YOUR HOST(S)

Manually gather logs in situations where **must-gather** or automated collection methods do not work.

Prerequisites

- You must have SSH access to your host(s).

Procedure

1. Collect the **bootkube.service** service logs from the bootstrap host using the **journalctl** command by running:

```
$ journalctl -b -f -u bootkube.service
```

2. Collect the bootstrap host's container logs using the podman logs. This is shown as a loop to get all of the container logs from the host:

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. Alternatively, collect the host's container logs using the **tail** command by running:

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. Collect the **kubelet.service** and **crio.service** service logs from the master and worker hosts using the **journalctl** command by running:

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. Collect the master and worker host container logs using the **tail** command by running:

```
$ sudo tail -f /var/log/containers/*
```

14.4. MANUALLY GATHERING LOGS WITHOUT SSH ACCESS TO YOUR HOST(S)

Manually gather logs in situations where **must-gather** or automated collection methods do not work.

If you do not have SSH access to your node, you can access the systems journal to investigate what is happening on your host.

Prerequisites

- Your OpenShift Container Platform installation must be complete.
- Your API service is still functional.
- You have system administrator privileges.

Procedure

1. Access **journald** unit logs under **/var/log** by running:

```
$ oc adm node-logs --role=master -u kubelet
```

2. Access host file paths under **/var/log** by running:

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

14.5. GETTING DEBUG INFORMATION FROM THE INSTALLATION PROGRAM

You can use any of the following actions to get debug information from the installation program.

- Look at debug messages from a past installation in the hidden **.openshift_install.log** file. For example, enter:

```
$ cat ~/<installation_directory>/.openshift_install.log 1
```

- 1** For **installation_directory**, specify the same directory you specified when you ran **./openshift-install create cluster**.

- Re-run the installation program with **--log-level=debug**:

```
$ ./openshift-install create cluster --dir=<installation_directory> --log-level=debug 1
```

-

- 1 For **installation_directory**, specify the same directory you specified when you ran **./openshift-install create cluster**.

CHAPTER 15. SUPPORT FOR FIPS CRYPTOGRAPHY

Starting with version 4.3, you can install an OpenShift Container Platform cluster that uses FIPS Validated / Modules in Process cryptographic libraries.

For the Red Hat Enterprise Linux CoreOS (RHCOS) machines in your cluster, this change is applied when the machines are deployed based on the status of an option in the **install-config.yaml** file, which governs the cluster options that a user can change during cluster deployment. With Red Hat Enterprise Linux (RHEL) machines, you must enable FIPS mode when you install the operating system on the machines that you plan to use as worker machines. These configuration methods ensure that your cluster meet the requirements of a FIPS compliance audit: only FIPS Validated / Modules in Process cryptography packages are enabled before the initial system boot.

Because FIPS must be enabled before the operating system that your cluster uses boots for the first time, you cannot enable FIPS after you deploy a cluster.

15.1. FIPS VALIDATION IN OPENSIFT CONTAINER PLATFORM

OpenShift Container Platform uses certain FIPS Validated / Modules in Process modules within RHEL and RHCOS for the operating system components that it uses. See [RHEL7 core crypto components](#). For example, when users SSH into OpenShift Container Platform clusters and containers, those connections are properly encrypted.

OpenShift Container Platform components are written in Go and built with Red Hat's golang compiler. When you enable FIPS mode for your cluster, all OpenShift Container Platform components that require cryptographic signing call RHEL and RHCOS cryptographic libraries.

Table 15.1. FIPS mode attributes and limitations in OpenShift Container Platform 4.5

Attributes	Limitations
FIPS support in RHEL 7 operating systems.	The FIPS implementation does not offer a single function that both computes hash functions and validates the keys that are based on that hash. This limitation will continue to be evaluated and improved in future OpenShift Container Platform releases.
FIPS support in CRI-O runtimes.	
FIPS support in OpenShift Container Platform services.	
FIPS Validated / Modules in Process cryptographic module and algorithms that are obtained from RHEL 7 and RHCOS binaries and images.	
Use of FIPS compatible golang compiler.	TLS FIPS support is not complete but is planned for future OpenShift Container Platform releases.

15.2. FIPS SUPPORT IN COMPONENTS THAT THE CLUSTER USES

Although the OpenShift Container Platform cluster itself uses FIPS Validated / Modules in Process modules, ensure that the systems that support your OpenShift Container Platform cluster use FIPS Validated / Modules in Process modules for cryptography.

15.2.1. etcd

To ensure that the secrets that are stored in etcd use FIPS Validated / Modules in Process encryption, boot the node in FIPS mode. After you install the cluster in FIPS mode, you can [encrypt the etcd data](#) by using the FIPS-approved **aes cbc** cryptographic algorithm.

15.2.2. Storage

For local storage, use RHEL-provided disk encryption or Container Native Storage that uses RHEL-provided disk encryption. By storing all data in volumes that use RHEL-provided disk encryption and enabling FIPS mode for your cluster, both data at rest and data in motion, or network data, are protected by FIPS Validated / Modules in Process encryption. You can configure your cluster to encrypt the root filesystem of each node, as described in [Customizing nodes](#).

15.2.3. Runtimes

To ensure that containers know that they are running on a host that is using FIPS Validated / Modules in Process cryptography modules, use CRI-O to manage your runtimes. CRI-O supports FIPS mode, in that it configures the containers to know that they are running in FIPS mode.

15.3. INSTALLING A CLUSTER IN FIPS MODE

To install a cluster in FIPS mode, follow the instructions to install a customized cluster on your preferred infrastructure. Ensure that you set **fips: true** in the **install-config.yaml** file before you deploy your cluster.

- [Amazon Web Services](#)
- [Microsoft Azure](#)
- [Bare metal](#)
- [Google Cloud Platform](#)
- [Red Hat OpenStack Platform \(RHOSP\)](#)
- [VMware vSphere](#)



NOTE

If you are using Azure File storage, you cannot enable FIPS mode.

To apply **AES CBC** encryption to your etcd data store, follow the [Encrypting etcd data](#) process after you install your cluster.

If you add RHEL nodes to your cluster, ensure that you enable FIPS mode on the machines before their initial boot. See [Adding RHEL compute machines to an OpenShift Container Platform cluster](#) and [Enabling FIPS Mode](#) in the RHEL 7 documentation.