OpenShift Container Platform 4.5

Installing on GCP

Installing OpenShift Container Platform GCP clusters
Installing OpenShift Container Platform GCP clusters
Abstract

This document provides instructions for installing and uninstalling OpenShift Container Platform clusters on Google Cloud Platform.
# Table of Contents

## CHAPTER 1. INSTALLING ON GCP

1.1. CONFIGURING A GCP PROJECT .......................... 8
  1.1.1. Creating a GCP project .......................... 8
  1.1.2. Enabling API services in GCP ....................... 8
  1.1.3. Configuring DNS for GCP .......................... 9
  1.1.4. GCP account limits ................................ 10
  1.1.5. Creating a service account in GCP ................. 11
    1.1.5.1. Required GCP permissions ...................... 12
  1.1.6. Supported GCP regions ............................ 13
  1.1.7. Next steps ...................................... 13

1.2. INSTALLING A CLUSTER QUICKLY ON GCP ............. 14
  1.2.1. Prerequisites .................................. 14
  1.2.2. Internet and Telemetry access for OpenShift Container Platform .......................... 14
  1.2.3. Generating an SSH private key and adding it to the agent .......................... 14
  1.2.4. Obtaining the installation program ................. 16
  1.2.5. Deploying the cluster ............................ 17
  1.2.6. Installing the CLI by downloading the binary .... 19
    1.2.6.1. Installing the CLI on Linux .................... 19
    1.2.6.2. Installing the CLI on Windows ................. 20
    1.2.6.3. Installing the CLI on macOS ................. 20
  1.2.7. Logging in to the cluster ........................ 21
  1.2.8. Next steps .................................... 21

1.3. INSTALLING A CLUSTER ON GCP WITH CUSTOMIZATIONS 21
  1.3.1. Prerequisites .................................. 21
  1.3.2. Internet and Telemetry access for OpenShift Container Platform .......................... 22
  1.3.3. Generating an SSH private key and adding it to the agent .......................... 22
  1.3.4. Obtaining the installation program ................. 24
  1.3.5. Creating the installation configuration file .... 24
    1.3.5.1. Installation configuration parameters ......... 26
      1.3.5.1.1. Required configuration parameters ........ 26
      1.3.5.1.2. Network configuration parameters .......... 28
      1.3.5.1.3. Optional configuration parameters ........ 29
      1.3.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters .......................... 33
    1.3.5.2. Sample customized install-config.yaml file for GCP .......................... 33
  1.3.6. Deploying the cluster ............................ 35
  1.3.7. Installing the CLI by downloading the binary .... 36
    1.3.7.1. Installing the CLI on Linux .................... 37
    1.3.7.2. Installing the CLI on Windows ................. 37
    1.3.7.3. Installing the CLI on macOS ................. 38
  1.3.8. Logging in to the cluster ........................ 38
  1.3.9. Next steps .................................... 39

1.4. INSTALLING A CLUSTER ON GCP WITH NETWORK CUSTOMIZATIONS 39
  1.4.1. Prerequisites .................................. 39
  1.4.2. Internet and Telemetry access for OpenShift Container Platform .......................... 39
  1.4.3. Generating an SSH private key and adding it to the agent .......................... 40
  1.4.4. Obtaining the installation program ................. 41
  1.4.5. Creating the installation configuration file .... 42
    1.4.5.1. Installation configuration parameters ......... 43
      1.4.5.1.1. Required configuration parameters ........ 44
      1.4.5.1.2. Network configuration parameters .......... 45
      1.4.5.1.3. Optional configuration parameters ........ 47
1.4.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters
1.4.5.2. Network configuration parameters
1.4.5.3. Sample customized install-config.yaml file for GCP
1.4.6. Modifying advanced network configuration parameters
1.4.7. Cluster Network Operator configuration
  1.4.7.1. Configuration parameters for the OpenShift SDN default CNI network provider
  1.4.7.2. Configuration parameters for the OVN-Kubernetes default CNI network provider
  1.4.7.3. Cluster Network Operator example configuration
1.4.8. Deploying the cluster
1.4.9. Installing the CLI by downloading the binary
  1.4.9.1. Installing the CLI on Linux
  1.4.9.2. Installing the CLI on Windows
  1.4.9.3. Installing the CLI on macOS
1.4.10. Logging in to the cluster
1.4.11. Next steps

1.5. INSTALLING A CLUSTER ON GCP INTO AN EXISTING VPC
1.5.1. Prerequisites
1.5.2. Internet and Telemetry access for OpenShift Container Platform
1.5.3. Generating an SSH private key and adding it to the agent
1.5.4. Obtaining the installation program
1.5.5. Creating the installation configuration file
  1.5.5.1. Installation configuration parameters
    1.5.5.1.1. Required configuration parameters
    1.5.5.1.2. Network configuration parameters
    1.5.5.1.3. Optional configuration parameters
    1.5.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters
  1.5.5.2. Sample customized install-config.yaml file for GCP
  1.5.5.3. Configuring the cluster-wide proxy during installation
1.5.6. Deploying the cluster
1.5.7. Installing the CLI by downloading the binary
  1.5.7.1. Installing the CLI on Linux
  1.5.7.2. Installing the CLI on Windows
  1.5.7.3. Installing the CLI on macOS
1.5.8. Logging in to the cluster
1.5.9. Next steps

1.6. INSTALLING A PRIVATE CLUSTER ON GCP
1.6.1. Prerequisites
1.6.2. Private clusters
  1.6.2.1. Private clusters in GCP
    1.6.2.1.1. Limitations
1.6.3. About using a custom VPC
  1.6.3.1. Requirements for using your VPC
  1.6.3.2. Division of permissions
  1.6.3.3. Isolation between clusters
1.6.4. Internet and Telemetry access for OpenShift Container Platform
1.6.5. Generating an SSH private key and adding it to the agent
1.6.6. Obtaining the installation program
1.6.7. Manually creating the installation configuration file
  1.6.7.1. Installation configuration parameters
    1.6.7.1.1. Required configuration parameters
    1.6.7.1.2. Network configuration parameters
    1.6.7.1.3. Optional configuration parameters
    1.6.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters
1.6.7.2. Sample customized install-config.yaml file for GCP
1.6.7.3. Configuring the cluster-wide proxy during installation
1.6.8. Deploying the cluster
1.6.9. Installing the CLI by downloading the binary

1.6.9.1. Installing the CLI on Linux
1.6.9.2. Installing the CLI on Windows
1.6.9.3. Installing the CLI on macOS
1.6.10. Logging in to the cluster
1.6.11. Next steps

1.7. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES

1.7.1. Prerequisites
1.7.2. Certificate signing requests management
1.7.3. Configuring your GCP project
  1.7.3.1. Creating a GCP project
  1.7.3.2. Enabling API services in GCP
  1.7.3.3. Configuring DNS for GCP
  1.7.3.4. GCP account limits
  1.7.3.5. Creating a service account in GCP
    1.7.3.5.1. Required GCP permissions
  1.7.3.6. Supported GCP regions
  1.7.3.7. Installing and configuring CLI tools for GCP
1.7.4. Creating the installation files for GCP
  1.7.4.1. Creating the installation configuration file
  1.7.4.2. Configuring the cluster-wide proxy during installation
  1.7.4.3. Creating the Kubernetes manifest and Ignition config files
1.7.5. Exporting common variables
  1.7.5.1. Extracting the infrastructure name
  1.7.5.2. Exporting common variables for Deployment Manager templates
1.7.6. Creating a VPC in GCP
  1.7.6.1. Deployment Manager template for the VPC
1.7.7. Networking requirements for user-provisioned infrastructure
  Network topology requirements
  Load balancers
1.7.8. Creating load balancers in GCP
  1.7.8.1. Deployment Manager template for the external load balancer
  1.7.8.2. Deployment Manager template for the internal load balancer
1.7.9. Creating a private DNS zone in GCP
  1.7.9.1. Deployment Manager template for the private DNS
1.7.10. Creating firewall rules in GCP
  1.7.10.1. Deployment Manager template for firewall rules
1.7.11. Creating IAM roles in GCP
  1.7.11.1. Deployment Manager template for IAM roles
1.7.12. Creating the RHCOS cluster image for the GCP infrastructure
1.7.13. Creating the bootstrap machine in GCP
  1.7.13.1. Deployment Manager template for the bootstrap machine
1.7.14. Creating the control plane machines in GCP
  1.7.14.1. Deployment Manager template for control plane machines
1.7.15. Wait for bootstrap completion and remove bootstrap resources in GCP
1.7.16. Creating additional worker machines in GCP
  1.7.16.1. Deployment Manager template for worker machines
1.7.17. Installing the CLI by downloading the binary
  1.7.17.1. Installing the CLI on Linux
1.7.17.2. Installing the CLI on Windows
1.7.17.3. Installing the CLI on macOS
1.7.18. Logging in to the cluster
1.7.19. Approving the certificate signing requests for your machines
1.7.20. Optional: Adding the ingress DNS records
1.7.21. Completing a GCP installation on user-provisioned infrastructure
1.7.22. Next steps

1.8. INSTALLING A CLUSTER WITH SHARED VPC ON USER-PROVISIONED INFRASTRUCTURE IN GCP BY USING DEPLOYMENT MANAGER TEMPLATES

1.8.1. Prerequisites
1.8.2. Certificate signing requests management
1.8.3. Configuring the GCP project that hosts your cluster
1.8.3.1. Creating a GCP project
1.8.3.2. Enabling API services in GCP
1.8.3.3. GCP account limits
1.8.3.4. Creating a service account in GCP
1.8.3.4.1. Required GCP permissions
1.8.3.5. Supported GCP regions
1.8.3.6. Installing and configuring CLI tools for GCP
1.8.4. Configuring the GCP project that hosts your shared VPC network
1.8.4.1. Configuring DNS for GCP
1.8.4.2. Creating a VPC in GCP
1.8.4.2.1. Deployment Manager template for the VPC
1.8.5. Creating the installation files for GCP
1.8.5.1. Manually creating the installation configuration file
1.8.5.2. Sample customized install-config.yaml file for GCP
1.8.5.3. Configuring the cluster-wide proxy during installation
1.8.5.4. Creating the Kubernetes manifest and Ignition config files
1.8.6. Exporting common variables
1.8.6.1. Extracting the infrastructure name
1.8.6.2. Exporting common variables for Deployment Manager templates
1.8.7. Networking requirements for user-provisioned infrastructure
Network topology requirements
Load balancers
1.8.8. Creating load balancers in GCP
1.8.8.1. Deployment Manager template for the external load balancer
1.8.8.2. Deployment Manager template for the internal load balancer
1.8.9. Creating a private DNS zone in GCP
1.8.9.1. Deployment Manager template for the private DNS
1.8.10. Creating firewall rules in GCP
1.8.10.1. Deployment Manager template for firewall rules
1.8.11. Creating IAM roles in GCP
1.8.11.1. Deployment Manager template for IAM roles
1.8.12. Creating the RHCOS cluster image for the GCP infrastructure
1.8.13. Creating the bootstrap machine in GCP
1.8.13.1. Deployment Manager template for the bootstrap machine
1.8.14. Creating the control plane machines in GCP
1.8.14.1. Deployment Manager template for control plane machines
1.8.15. Wait for bootstrap completion and remove bootstrap resources in GCP
1.8.16. Creating additional worker machines in GCP
1.8.16.1. Deployment Manager template for worker machines
1.8.17. Installing the CLI by downloading the binary
1.8.17.1. Installing the CLI on Linux
1.8.17.2. Installing the CLI on Windows
1.8.17.3. Installing the CLI on macOS
1.8.18. Logging in to the cluster
1.8.19. Approving the certificate signing requests for your machines
1.8.20. Adding the ingress DNS records
1.8.21. Adding ingress firewall rules
1.8.21.1. Creating cluster-wide firewall rules for a shared VPC in GCP
1.8.22. Completing a GCP installation on user-provisioned infrastructure
1.8.23. Next steps

1.9. INSTALLING A CLUSTER ON GCP IN A RESTRICTED NETWORK WITH USER-PROVISIONED INFRASTRUCTURE

1.9.1. Prerequisites
1.9.2. Configuring your GCP project
1.9.2.1. Creating a GCP project
1.9.2.2. Enabling API services in GCP
1.9.2.3. Configuring DNS for GCP
1.9.2.4. GCP account limits
1.9.2.5. Creating a service account in GCP
1.9.2.5.1. Required GCP permissions
1.9.2.6. Supported GCP regions
1.9.2.7. Installing and configuring CLI tools for GCP
1.9.3. Creating the installation files for GCP
1.9.3.1. Creating the installation configuration file
1.9.3.2. Creating the Kubernetes manifest and Ignition config files
1.9.4. Exporting common variables
1.9.4.1. Extracting the infrastructure name
1.9.4.2. Exporting common variables for Deployment Manager templates
1.9.5. Creating a VPC in GCP
1.9.5.1. Deployment Manager template for the VPC
1.9.6. Networking requirements for user-provisioned infrastructure
   Network topology requirements
   Load balancers
1.9.7. Creating load balancers in GCP
1.9.7.1. Deployment Manager template for the external load balancer
1.9.7.2. Deployment Manager template for the internal load balancer
1.9.8. Creating a private DNS zone in GCP
1.9.8.1. Deployment Manager template for the private DNS
1.9.9. Creating firewall rules in GCP
1.9.9.1. Deployment Manager template for firewall rules
1.9.10. Creating IAM roles in GCP
1.9.10.1. Deployment Manager template for IAM roles
1.9.11. Creating the RH COS cluster image for the GCP infrastructure
1.9.12. Creating the bootstrap machine in GCP
1.9.12.1. Deployment Manager template for the bootstrap machine
1.9.13. Creating the control plane machines in GCP
1.9.13.1. Deployment Manager template for control plane machines
1.9.14. Wait for bootstrap completion and remove bootstrap resources in GCP
1.9.15. Creating additional worker machines in GCP
1.9.15.1. Deployment Manager template for worker machines
1.9.16. Logging in to the cluster
1.9.17. Approving the certificate signing requests for your machines
1.9.18. Optional: Adding the ingress DNS records
1.9.19. Completing a GCP installation on user-provisioned infrastructure
1.9.20. Next steps

1.10. UNINSTALLING A CLUSTER ON GCP

1.10.1. Removing a cluster that uses installer-provisioned infrastructure
CHAPTER 1. INSTALLING ON GCP

1.1. CONFIGURING A GCP PROJECT

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

1.1.1. 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.

1.1.2. 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 1.1. Required API services

<table>
<thead>
<tr>
<th>API service</th>
<th>Console service name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Engine API</td>
<td>compute.googleapis.com</td>
</tr>
<tr>
<td>Google Cloud APIs</td>
<td>cloudapis.googleapis.com</td>
</tr>
<tr>
<td>Cloud Resource Manager API</td>
<td>cloudresourcemanager.googleapis.com</td>
</tr>
<tr>
<td>Google DNS API</td>
<td>dns.googleapis.com</td>
</tr>
<tr>
<td>API service</td>
<td>Console service name</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>IAM Service Account Credentials API</td>
<td>iamcredentials.googleapis.com</td>
</tr>
<tr>
<td>Identity and Access Management (IAM) API</td>
<td>iam.googleapis.com</td>
</tr>
<tr>
<td>Service Management API</td>
<td>servicemanagement.googleapis.com</td>
</tr>
<tr>
<td>Service Usage API</td>
<td>serviceusage.googleapis.com</td>
</tr>
<tr>
<td>Google Cloud Storage JSON API</td>
<td>storage-api.googleapis.com</td>
</tr>
<tr>
<td>Cloud Storage</td>
<td>storage-component.googleapis.com</td>
</tr>
</tbody>
</table>

### 1.1.3. 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](https://domains.google.com/).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](https://cloud.google.com/dns/docs/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](https://cloud.google.com/dns/docs/authoritative-nameservers) 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](https://domains.google.com/).

5. If you migrated your root domain to Google Cloud DNS, migrate your DNS records. See [Migrating to Cloud DNS](https://cloud.google.com/dns/docs/migrate) 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.

1.1.4. 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 1.2. GCP resources used in a default cluster**

<table>
<thead>
<tr>
<th>Service</th>
<th>Component</th>
<th>Location</th>
<th>Total resources required</th>
<th>Resources removed after bootstrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service account</td>
<td>IAM</td>
<td>Global</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Firewall rules</td>
<td>Compute</td>
<td>Global</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>In-use global IP addresses</td>
<td>Compute</td>
<td>Global</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Health checks</td>
<td>Compute</td>
<td>Global</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Images</td>
<td>Compute</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Networks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Static IP addresses</td>
<td>Compute</td>
<td>Region</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Routers</td>
<td>Compute</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routes</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Subnetworks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Target pools</td>
<td>Compute</td>
<td>Global</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>CPUs</td>
<td>Compute</td>
<td>Region</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>Persistent disk SSD (GB)</td>
<td>Compute</td>
<td>Region</td>
<td>896</td>
<td>128</td>
</tr>
</tbody>
</table>
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
- europe-north1
- europe-west2
- europe-west3
- europe-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.

1.1.5. 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.

1.1.5.1. 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 1.3. GCP service account permissions

<table>
<thead>
<tr>
<th>Account</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane</td>
<td><code>roles/compute.instanceAdmin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/compute.networkAdmin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/compute.securityAdmin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/storage.admin</code></td>
</tr>
</tbody>
</table>
1.1.6. 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)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-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)

1.1.7. Next steps
1.2. 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.

1.2.1. 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.

1.2.2. 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.

1.2.3. 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:

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

   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:

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

   **Example output**

   ```bash
   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`:

      ```bash
      $ ssh-add <path>/<file_name>
      ```
Example output

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

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.

1.2.4. 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.

### 1.2.5. 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 `GCLOUD_KEYFILE_JSON` environment variables
   - The `~/.gcp/osServiceAccount.json` file
   - The `gcloud cli` default credentials

2. Run the installation program:

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

1.2.6. 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**.

1.2.6.1. 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 xzvf <file>
   ```

5. Place the **oc** binary in a directory that is on your **PATH**.

   To check your **PATH**, execute the following command:
After you install the CLI, it is available using the `oc` command:

```
$ echo $PATH
```

1.2.6.2. 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>
```

1.2.6.3. 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:
1.2.7. 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
   ```

   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
   ```

1.2.8. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting.

1.3. 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.

1.3.1. 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.

1.3.2. 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.

1.3.3. 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:

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

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:

```bash
$ 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`:

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

**Example output**

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

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.

```bash
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```bash
$ gcloud auth list
```
Next steps

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

1.3.4. 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.

1.3.5. 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:

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

   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.

   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.

1.3.5.1. 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.

1.3.5.1.1. Required configuration parameters
Required installation configuration parameters are described in the following table:

Table 1.4. Required parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>The API version for the install-config.yaml content. The current version is v1. The installer may also support older API versions.</td>
<td>String</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>baseDomain</td>
<td>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 &lt;metadata.name&gt;, &lt;baseDomain&gt; format.</td>
<td>A fully-qualified domain or subdomain name, such as example.com.</td>
</tr>
<tr>
<td>metadata</td>
<td>Kubernetes resource ObjectMeta, from which only the name parameter is consumed.</td>
<td>Object</td>
</tr>
<tr>
<td>metadata.name</td>
<td>The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}, {{.baseDomain}}.</td>
<td>String of lowercase letters, hyphens (-), and periods (.), such as dev.</td>
</tr>
<tr>
<td>platform</td>
<td>The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere. For additional information about platform.&lt;platform&gt; parameters, consult the following table for your specific platform.</td>
<td>Object</td>
</tr>
</tbody>
</table>
### 1.3.5.1.2. 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 1.5. Network parameters

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

#### networking

The configuration for the cluster network.

- **Type**: Object

  **NOTE**
  
  You cannot modify parameters specified by the `networking` object after installation.

  ```
  networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
  hostPrefix: 23
  ```

  
  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.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>networking.clusterNetwork.cidr</code></td>
<td>Required if you use <code>networking.clusterNetwork</code>. An IP address block. An IPv4 network.</td>
<td>An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32.</td>
</tr>
<tr>
<td><code>networking.clusterNetwork.hostPrefix</code></td>
<td>The subnet prefix length to assign to each individual node. For example, if <code>hostPrefix</code> is set to 23 then each node is assigned a /23 subnet out of the given <code>cidr</code>. A <code>hostPrefix</code> value of 23 provides 510 (2^(32 - 23) - 2) pod IP addresses.</td>
<td>A subnet prefix. The default value is 23.</td>
</tr>
<tr>
<td><code>networking.serviceNetwork</code></td>
<td>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.</td>
<td>An array with an IP address block in CIDR format. For example: networking: serviceNetwork: - 172.30.0.0/16</td>
</tr>
<tr>
<td><code>networking.machineNetwork</code></td>
<td>The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.</td>
<td>An array of objects. For example: networking: machineNetwork: - cidr: 10.0.0.0/16</td>
</tr>
<tr>
<td><code>networking.machineNetwork.cidr</code></td>
<td>Required if you use <code>networking.machineNetwork</code>. 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.</td>
<td>An IP network block in CIDR notation. For example, 10.0.0.0/16. <strong>NOTE</strong> Set the <code>networking.machineNetwork</code> to match the CIDR that the preferred NIC resides in.</td>
</tr>
</tbody>
</table>

1.3.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.6. Optional parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalTrustBundle</td>
<td>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.</td>
<td>String</td>
</tr>
<tr>
<td>compute</td>
<td>The configuration for the machines that comprise the compute nodes.</td>
<td>Array of machine-pool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>compute.architecture</td>
<td>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 <strong>amd64</strong> (the default).</td>
<td>String</td>
</tr>
<tr>
<td>compute.hyperthreading</td>
<td>Whether to enable or disable simultaneous multithreading, or <strong>hyperthreading</strong>, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td></td>
<td><strong>IMPORTANT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</td>
<td></td>
</tr>
<tr>
<td>compute.name</td>
<td>Required if you use <strong>compute</strong>. The name of the machine pool.</td>
<td>worker</td>
</tr>
<tr>
<td>compute.platform</td>
<td>Required if you use <strong>compute</strong>. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the <strong>controlPlane.platform</strong> parameter value.</td>
<td>aws, azure, gcp, openstack, ovirt, vsphere, or{}</td>
</tr>
<tr>
<td>compute.replicas</td>
<td>The number of compute machines, which are also known as worker machines, to provision.</td>
<td>A positive integer greater than or equal to 2. The default value is 3.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>controlPlane</td>
<td>The configuration for the machines that comprise the control plane.</td>
<td>Array of MachinePool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>controlPlane.architecture</td>
<td>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 <strong>amd64</strong> (the default).</td>
<td>String</td>
</tr>
<tr>
<td>controlPlane.hypertreading</td>
<td>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.</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td>controlPlane.name</td>
<td>Required if you use <strong>controlPlane</strong>. The name of the machine pool.</td>
<td>master</td>
</tr>
<tr>
<td>controlPlane.platform</td>
<td>Required if you use <strong>controlPlane</strong>. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the <strong>compute.platform</strong> parameter value.</td>
<td>aws, azure, gcp, openstack, ovirt, vsphere, or {}</td>
</tr>
<tr>
<td>controlPlane.replicas</td>
<td>The number of control plane machines to provision.</td>
<td>The only supported value is 3, which is the default value.</td>
</tr>
</tbody>
</table>
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.

**NOTE**

If you are using Azure File storage, you cannot enable FIPS mode.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>fips</td>
<td>Enable or disable FIPS mode.</td>
<td><code>false</code> or <code>true</code></td>
</tr>
<tr>
<td>imageContentSources</td>
<td>Sources and repositories for the release-image content.</td>
<td>Array of objects. Includes a <code>source</code> and, optionally, <code>mirrors</code>, as described in the following rows of this table.</td>
</tr>
<tr>
<td>imageContentSources.source</td>
<td>Required if you use <code>imageContentSources</code>. Specify the repository that users refer to, for example, in image pull specifications.</td>
<td>String</td>
</tr>
<tr>
<td>imageContentSources.mirrors</td>
<td>Specify one or more repositories that may also contain the same images.</td>
<td>Array of strings</td>
</tr>
<tr>
<td>publish</td>
<td>How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.</td>
<td><code>Internal</code> or <code>External</code>. To deploy a private cluster, which cannot be accessed from the internet, set <code>publish</code> to <code>Internal</code>. The default value is <code>External</code>.</td>
</tr>
</tbody>
</table>
The SSH key to authenticate access to 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. For example, `sshKey: ssh-ed25519 AAAA...`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sshKey</strong></td>
<td>The SSH key to authenticate access to your cluster machines.</td>
<td>For example, <code>sshKey: ssh-ed25519 AAAA...</code></td>
</tr>
</tbody>
</table>

### 1.3.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

**Table 1.7. Additional GCP parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>platform.gcp.network</code></td>
<td>The name of the existing VPC that you want to deploy your cluster to.</td>
<td>String.</td>
</tr>
<tr>
<td><code>platform.gcp.type</code></td>
<td>The GCP machine type.</td>
<td>The GCP machine type.</td>
</tr>
<tr>
<td><code>platform.gcp.zones</code></td>
<td>The availability zones where the installation program creates machines for the specified MachinePool.</td>
<td>A list of valid GCP availability zones, such as <code>us-central1-a</code>, in a YAML sequence.</td>
</tr>
<tr>
<td><code>platform.gcp.controlPlaneSubnet</code></td>
<td>The name of the existing subnet in your VPC that you want to deploy your control plane machines to.</td>
<td>The subnet name.</td>
</tr>
<tr>
<td><code>platform.gcp.computeSubnet</code></td>
<td>The name of the existing subnet in your VPC that you want to deploy your compute machines to.</td>
<td>The subnet name.</td>
</tr>
</tbody>
</table>

### 1.3.5.2. 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.

---

33
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.

```yaml
apiVersion: v1
baseDomain: example.com
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
      replicas: 3
  compute:
    - hyperthreading: Enabled
      name: worker
      platform:
        gcp:
          type: n2-standard-4
          zones:
            - us-central1-a
            - us-central1-c
          replicas: 3
  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
      region: us-central1
      pullSecret: '{"auths": ...}'
      fips: false
      sshKey: ssh-ed25519 AAAA...
```

1. `apiVersion: v1` - Required. The installation program prompts you for this value.
2. `baseDomain: example.com` - If you do not provide these parameters and values, the installation program provides the default value.
3. `controlPlane:` - 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

---

OpenShift Container Platform 4.5 Installing on GCP
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.

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.

1.3.6. 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 `GCloud_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> \  
   --log-level=info

   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.

**1.3.7. 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.
1.3.7.1. 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>
```

1.3.7.2. 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:
1.3.7.3. 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>
```

1.3.8. 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
   ```

   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
   ```
1.3.9. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting.

1.4. 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.

1.4.1. 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.

1.4.2. 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.

1.4.3. 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:

   ```bash
   $ ssh-keygen -t ed25519 -N "\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:

```bash
$ 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**:

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

**Example output**

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

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.

```bash
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. Verify that the credentials were applied.

```bash
$ gcloud auth list
```

**Next steps**

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

**1.4.4. 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.

### 1.4.5. 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>
   ```

   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.

1.4.5.1. 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.

### 1.4.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>apiVersion</strong></td>
<td>The API version for the <code>install-config.yaml</code> content. The current version is <code>v1</code>. The installer may also support older API versions.</td>
<td>String</td>
</tr>
<tr>
<td><strong>baseDomain</strong></td>
<td>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 <code>baseDomain</code> and <code>metadata.name</code> parameter values that uses the <code>&lt;metadata.name&gt;</code>, <code>&lt;baseDomain&gt;</code> format.</td>
<td>A fully-qualified domain or subdomain name, such as <code>example.com</code>.</td>
</tr>
<tr>
<td><strong>metadata</strong></td>
<td>Kubernetes resource <code>ObjectMeta</code>, from which only the <code>name</code> parameter is consumed.</td>
<td>Object</td>
</tr>
<tr>
<td><strong>metadata.name</strong></td>
<td>The name of the cluster. DNS records for the cluster are all subdomains of <code>{{.metadata.name}}.{{.baseDomain}}</code>.</td>
<td>String of lowercase letters, hyphens (-), and periods (.), such as <code>dev</code>.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>platform</strong></td>
<td>The configuration for the specific platform upon which to perform the installation: <strong>aws, baremetal, azure, openstack, ovirt, vsphere.</strong> For additional information about <code>platform.&lt;platform&gt;</code> parameters, consult the following table for your specific platform.</td>
<td>Object</td>
</tr>
</tbody>
</table>

**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.

```json
{
  "auths":{
    "cloud.openshift.com":{
      "auth":"b3Blb=",
      "email":"you@example.com"
    },
    "quay.io":{
      "auth":"b3Blb=",
      "email":"you@example.com"
    }
  }
}
```

### 1.4.5.1.2. 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 1.9. Network parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>networking</strong></td>
<td>The configuration for the cluster network.</td>
<td>Object</td>
</tr>
</tbody>
</table>

**NOTE**

You cannot modify parameters specified by the `networking` object after installation.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>networking.clusterNetwork</td>
<td>The IP address blocks for pods. The default value is 10.128.0.0/14 with a host prefix of /23.</td>
<td>An array of objects. For example:</td>
</tr>
<tr>
<td></td>
<td>If you specify multiple IP address blocks, the blocks must not overlap.</td>
<td>networking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clusterNetwork:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- cidr: 10.128.0.0/14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- hostPrefix: 23</td>
</tr>
<tr>
<td>networking.clusterNetwork.cidr</td>
<td>Required if you use networking.clusterNetwork. An IP address block.</td>
<td>An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32.</td>
</tr>
<tr>
<td></td>
<td>An IPv4 network.</td>
<td>networking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>serviceNetwork:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 172.30.0.0/16</td>
</tr>
<tr>
<td>networking.clusterNetwork.hostPrefix</td>
<td>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.</td>
<td>A subnet prefix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default value is 23</td>
</tr>
<tr>
<td>networking.serviceNetwork</td>
<td>The IP address block for services. The default value is 172.30.0.0/16.</td>
<td>An array with an IP address block in CIDR format. For example:</td>
</tr>
<tr>
<td></td>
<td>The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network.</td>
<td>networking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>serviceNetwork:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 172.30.0.0/16</td>
</tr>
<tr>
<td>networking.machineNetwork</td>
<td>The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.</td>
<td>An array of objects. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>networking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machineNetwork:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- cidr: 10.0.0.0/16</td>
</tr>
</tbody>
</table>
### 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.

#### 1.4.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

**Table 1.10. Optional parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>additionalTrustBundle</strong></td>
<td>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.</td>
<td>String</td>
</tr>
<tr>
<td><strong>compute</strong></td>
<td>The configuration for the machines that comprise the compute nodes.</td>
<td>Array of machine-pool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td><strong>compute.architecture</strong></td>
<td>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 <code>amd64</code> (the default).</td>
<td>String</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute.hyperthreading</td>
<td>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.</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td>compute.name</td>
<td>Required if you use compute. The name of the machine pool.</td>
<td>worker</td>
</tr>
<tr>
<td>compute.platform</td>
<td>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.</td>
<td>aws, azure, gcp, openstack, ovirt, vsphere, or{}</td>
</tr>
<tr>
<td>compute.replicas</td>
<td>The number of compute machines, which are also known as worker machines, to provision.</td>
<td>A positive integer greater than or equal to 2. The default value is 3.</td>
</tr>
<tr>
<td>controlPlane</td>
<td>The configuration for the machines that comprise the control plane.</td>
<td>Array of MachinePool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>controlPlane.architecture</td>
<td>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).</td>
<td>String</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><code>controlPlane.hypertreading</code></td>
<td>Whether to enable or disable simultaneous multithreading, or <strong>hyperthreading</strong>, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines’ cores.</td>
<td><strong>Enabled or Disabled</strong></td>
</tr>
<tr>
<td><strong>IMPORTANT</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</td>
<td></td>
</tr>
<tr>
<td><code>controlPlane.name</code></td>
<td>Required if you use <code>controlPlane</code>.</td>
<td><code>master</code></td>
</tr>
<tr>
<td></td>
<td>The name of the machine pool.</td>
<td></td>
</tr>
<tr>
<td><code>controlPlane.platform</code></td>
<td>Required if you use <code>controlPlane</code>.</td>
<td><code>aws, azure, gcp, openstack, ovirt, vsphere, or {}</code></td>
</tr>
<tr>
<td></td>
<td>Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the <code>compute.platform</code> parameter value.</td>
<td></td>
</tr>
<tr>
<td><code>controlPlane.replicas</code></td>
<td>The number of control plane machines to provision.</td>
<td>The only supported value is <code>3</code>, which is the default value.</td>
</tr>
<tr>
<td><code>fips</code></td>
<td>Enable or disable FIPS mode. The default is <strong>false</strong> (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.</td>
<td><strong>false or true</strong></td>
</tr>
<tr>
<td><strong>NOTE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you are using Azure File storage, you cannot enable FIPS mode.</td>
<td></td>
</tr>
</tbody>
</table>
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.

Required if you use `imageContentSources`. Specify the repository that users refer to, for example, in image pull specifications.

Specifying one or more repositories that may also contain the same images.

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.

The SSH key to authenticate access to your cluster machines.

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.

### 1.4.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>platform.gcp.network</code></td>
<td>The name of the existing VPC that you want to deploy your cluster to.</td>
<td>String.</td>
</tr>
</tbody>
</table>
### 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](https://access.redhat.com/articles/4380121).

### 1.4.5.2. 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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>networking.networkType</td>
<td>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.</td>
<td>Either OpenShiftSDN or OVNKubernetes. The default value is OpenShiftSDN.</td>
</tr>
</tbody>
</table>
**networking.clusternetwork[].cidr**
A block of IP addresses from which pod IP addresses are allocated. The OpenShift SDN 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.

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>An IP address allocation in CIDR format. The default value is <strong>10.128.0.0/14</strong>.</td>
</tr>
</tbody>
</table>

**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 - 2^3) pod IP addresses.

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A subnet prefix. The default value is <strong>23</strong>.</td>
</tr>
</tbody>
</table>

**networking.servicenetwork[]**
A block of IP addresses for services. OpenShift SDN allows only one servicenetwork block. The address block must not overlap with any other network block.

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>An IP address allocation in CIDR format. The default value is <strong>172.30.0.0/16</strong>.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>An IP address allocation in CIDR format. The default value is <strong>10.0.0.0/16</strong>.</td>
</tr>
</tbody>
</table>

---

### 1.4.5.3. 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.

```yaml
apiVersion: v1
baseDomain: example.com
controlPlane: 2 3
  hyperthreading: Enabled 4
name: master
platform:
gcp:
  type: n2-standard-4
  zones:
    - us-central1-a
    - us-central1-c
replicas: 3 6
compute: 5
  - hyperthreading: Enabled 7
```
name: worker
platform:
gcp:
  type: n2-standard-4
zones:
  - us-central1-a
  - us-central1-c
replicas: 3
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
  region: us-central1
pullSecret: '{"auths": ...}'
fips: false
sshKey: ssh-ed25519 AAAA...

1 Required. The installation program prompts you for this value.
2 If you do not provide these parameters and values, the installation program provides the default value.
3 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 machine types, such as n1-standard-8, for your machines if you disable simultaneous multithreading.

5 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
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.

1.4.6. 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
   
   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
   
   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
3. Open the `cluster-network-03-config.yml` file in an editor and enter a CR that describes the Operator configuration you want:

```yaml
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
```

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.

### 1.4.7. 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

```yaml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
```
Specified in the install-config.yaml file.

Configures the default Container Network Interface (CNI) network provider for the cluster network.

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.

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.

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.

1.4.7.1. 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
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789

Specified in the install-config.yaml file.

Specify only if you want to override part of the OpenShift SDN configuration.

Configures the network isolation mode for OpenShift SDN. The allowed values are Multitenant, Subnet, or NetworkPolicy. The default value is NetworkPolicy.
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.

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.

1.4.7.2. 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
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
```

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.

1.4.7.3. Cluster Network Operator example configuration
A complete CR object for the CNO is displayed in the following example:

**Cluster Network Operator example CR**

```yaml
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
```

**1.4.8. 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:

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

   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.

### 1.4.9. 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**.

#### 1.4.9.1. 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 xzvf <file>
   ```
5. Place the **oc** binary in a directory that is on your **PATH**.

To check your **PATH**, execute the following command:
After you install the CLI, it is available using the `oc` command:

$ oc <command>

1.4.9.2. 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>

1.4.9.3. 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:
1.4.10. 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

   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

1.4.11. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting.

1.5. 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.

1.5.1. 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.

1.5.2. 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.

1.5.3. 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 "\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>)
   ```

   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.

1.5.4. 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.

1.5.5. 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:

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

   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.

### 1.5.5.1. 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.

### 1.5.5.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

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

### 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 1.14. Network parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>networking</td>
<td>The configuration for the cluster network.</td>
<td>Object</td>
</tr>
<tr>
<td>networking.network</td>
<td>The cluster network provider Container Network Interface (CNI) plug-in to install.</td>
<td>Either OpenShiftSDN or OVNKubernetes. The default value is OpenShiftSDN.</td>
</tr>
<tr>
<td>networking.clusterNetwork</td>
<td>The IP address blocks for pods.</td>
<td>An array of objects. For example:</td>
</tr>
</tbody>
</table>

<p>| networking.clusterNetwork                | The default value is 10.128.0.0/14 with a host prefix of /23. | networking: clusterNetwork: |
|                                          | If you specify multiple IP address blocks, the blocks must not overlap. | - cidr: 10.128.0.0/14 hostPrefix: 23 |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>networking.clusterNetwork.cidr</td>
<td>Required if you use networking.clusterNetwork. An IP address block.</td>
<td>An IP address block in Classless Inter-Domain Routing (CIDR notation). The prefix length for an IPv4 block is between 0 and 32.</td>
</tr>
<tr>
<td>networking.clusterNetwork.hostPrefix</td>
<td>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.</td>
<td>A subnet prefix. The default value is 23.</td>
</tr>
<tr>
<td>networking.serviceNetwork</td>
<td>The IP address block for services. The default value is 172.30.0.0/16.</td>
<td>An array with an IP address block in CIDR format. For example:</td>
</tr>
</tbody>
</table>
|                                   | The OpenShift SDN and OVN-Kubernetes network providers support only a single IP address block for the service network. | networking:
|                                   | serviceNetwork:                                                           | - 172.30.0.0/16                                                      |
| networking.machineNetwork         | The IP address blocks for machines.                                         | An array of objects. For example:                                     |
|                                   | If you specify multiple IP address blocks, the blocks must not overlap.     | networking:
|                                   |                                                                           | machineNetwork:                                                      |
|                                   |                                                                           | - cidr: 10.0.0.0/16                                                  |
| networking.machineNetwork.cidr    | Required if you use networking.machineNetwork. An IP address block.         | An IP network block in CIDR notation. For example, 10.0.0.0/16.       |
|                                   | 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. | For example, 10.0.0.0/16.                                             |
|                                   |                                                                           | NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in. |

1.5.5.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.15. Optional parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalTrustBundle</td>
<td>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.</td>
<td>String</td>
</tr>
<tr>
<td>compute</td>
<td>The configuration for the machines that comprise the compute nodes.</td>
<td>Array of machine-pool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>compute.architecture</td>
<td>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 <code>amd64</code> (the default).</td>
<td>String</td>
</tr>
<tr>
<td>compute.hyperthreading</td>
<td>Whether to enable or disable simultaneous multithreading, or <strong>hyperthreading</strong>, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td></td>
<td><strong>IMPORTANT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</td>
<td></td>
</tr>
<tr>
<td>compute.name</td>
<td>Required if you use <code>compute</code>. The name of the machine pool.</td>
<td>worker</td>
</tr>
<tr>
<td>compute.platform</td>
<td>Required if you use <code>compute</code>. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the <code>controlPlane.platform</code> parameter value.</td>
<td><code>aws</code>, <code>azure</code>, <code>gcp</code>, <code>openstack</code>, <code>ovirt</code>, <code>vsphere</code>, or <code>{}</code></td>
</tr>
<tr>
<td>compute.replicas</td>
<td>The number of compute machines, which are also known as worker machines, to provision.</td>
<td>A positive integer greater than or equal to 2. The default value is 3.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>controlPlane</td>
<td>The configuration for the machines that comprise the control plane.</td>
<td>Array of MachinePool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>controlPlane.architecture</td>
<td>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).</td>
<td>String</td>
</tr>
<tr>
<td>controlPlane.hypertreading</td>
<td>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.</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td>controlPlane.name</td>
<td>Required if you use controlPlane. The name of the machine pool.</td>
<td>master</td>
</tr>
<tr>
<td>controlPlane.platform</td>
<td>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.</td>
<td>aws, azure, gcp, openstack, ovirt, vsphere, or {}</td>
</tr>
<tr>
<td>controlPlane.replicas</td>
<td>The number of control plane machines to provision.</td>
<td>The only supported value is 3, which is the default value.</td>
</tr>
</tbody>
</table>
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.

**NOTE**
If you are using Azure File storage, you cannot enable FIPS mode.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fips</strong></td>
<td>Enable or disable FIPS mode. The default is <strong>false</strong> (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.</td>
<td><strong>false</strong> or <strong>true</strong></td>
</tr>
<tr>
<td><strong>imageContentSources</strong></td>
<td>Sources and repositories for the release-image content.</td>
<td>Array of objects. Includes a <strong>source</strong> and, optionally, <strong>mirrors</strong>, as described in the following rows of this table.</td>
</tr>
<tr>
<td><strong>imageContentSources.source</strong></td>
<td>Required if you use <strong>imageContentSources</strong>. Specify the repository that users refer to, for example, in image pull specifications.</td>
<td><strong>String</strong></td>
</tr>
<tr>
<td><strong>imageContentSources.mirrors</strong></td>
<td>Specify one or more repositories that may also contain the same images.</td>
<td>Array of strings</td>
</tr>
<tr>
<td><strong>publish</strong></td>
<td>How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.</td>
<td><strong>Internal</strong> or <strong>External</strong>. To deploy a private cluster, which cannot be accessed from the internet, set <strong>publish</strong> to <strong>Internal</strong>. The default value is <strong>External</strong>.</td>
</tr>
</tbody>
</table>
The SSH key to authenticate access to 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.

For example, `sshKey: ssh-ed25519 AAAA...`

### 1.5.5.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

**Table 1.16. Additional GCP parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>platform.gcp.network</code></td>
<td>The name of the existing VPC that you want to deploy your cluster to.</td>
<td>String.</td>
</tr>
<tr>
<td><code>platform.gcp.type</code></td>
<td>The GCP machine type.</td>
<td>The GCP machine type.</td>
</tr>
<tr>
<td><code>platform.gcp.zones</code></td>
<td>The availability zones where the installation program creates machines for the specified MachinePool.</td>
<td>A list of valid GCP availability zones, such as <code>us-central1-a</code>, in a YAML sequence.</td>
</tr>
<tr>
<td><code>platform.gcp.controlPlaneSubnet</code></td>
<td>The name of the existing subnet in your VPC that you want to deploy your control plane machines to.</td>
<td>The subnet name.</td>
</tr>
<tr>
<td><code>platform.gcp.computeSubnet</code></td>
<td>The name of the existing subnet in your VPC that you want to deploy your compute machines to.</td>
<td>The subnet name.</td>
</tr>
</tbody>
</table>

### 1.5.5.2. 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.
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.

```yaml
apiVersion: v1
baseDomain: example.com
controlPlane:
  hypethreading: Enabled
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
      replicas: 3
      compute:
        - hypethreading: Enabled
        name: worker
        platform:
          gcp:
            type: n2-standard-4
            zones:
            - us-central1-a
            - us-central1-c
            replicas: 3
        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
      region: us-central1
      network: existing_vpc
      controlPlaneSubnet: control_plane_subnet
      computeSubnet: compute_subnet
      pullSecret: '{"auths": ...}'
      fips: false
      sshKey: ssh-ed25519 AAAA...

Required. The installation program prompts you for this value.

If you do not provide these parameters and values, the installation program provides the default value.
3. 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. 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.

### 1.5.5.3. 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:

```yaml
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: http://<username>:<pswd>@<ip>:<port>
  noProxy: example.com
additionalTrustBundle: |
    -----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.

1.5.6. 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 `GCLOUD_KEYFILE_JSON` environment variables
   - The `~/.gcp/osServiceAccount.json` file
   - The `gcloud` cli default credentials

2. Run the installation program:

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

1.5.7. 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**.

1.5.7.1. 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>
```

### 1.5.7.2. 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. 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>
   ```

### 1.5.8. 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
   ```

   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
   ```

**1.5.9. Next steps**

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

### 1.6. 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.

### 1.6.1. 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.

### 1.6.2. 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.

#### 1.6.2.1. 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.

1.6.2.1. 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.

1.6.3. 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.

1.6.3.1. 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.

1.6.3.2. 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.

1.6.3.3. 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

1.6.4. 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.

### 1.6.5. 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:

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

   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>
      ```

      **Example output**

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

      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.

**1.6.6. 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
   $ mkdir <installation_directory>
   ```

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.

**1.6.7. 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.

1.6.7.1. 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.

1.6.7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 1.17. Required parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>apiVersion</td>
<td>The API version for the install-config.yaml content. The current version is v1. The installer may also support older API versions.</td>
<td>String</td>
</tr>
<tr>
<td>baseDomain</td>
<td>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 &lt;metadata.name&gt;, &lt;baseDomain&gt; format.</td>
<td>A fully-qualified domain or subdomain name, such as example.com.</td>
</tr>
<tr>
<td>metadata</td>
<td>Kubernetes resource ObjectMeta, from which only the name parameter is consumed.</td>
<td>Object</td>
</tr>
<tr>
<td>metadata.name</td>
<td>The name of the cluster. DNS records for the cluster are all subdomains of {{metadata.name}}. {{baseDomain}}.</td>
<td>String of lowercase letters, hyphens (-), and periods (.), such as dev.</td>
</tr>
<tr>
<td>platform</td>
<td>The configuration for the specific platform upon which to perform the installation: aws, baremetal, azure, openstack, ovirt, vsphere. For additional information about platform.&lt;platform&gt; parameters, consult the following table for your specific platform.</td>
<td>Object</td>
</tr>
</tbody>
</table>
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.

```json
{
  "auths":{
    "cloud.openshift.com":{
      "auth":"b3Blb=",
      "email":"you@example.com"
    },
    "quay.io":{
      "auth":"b3Blb=",
      "email":"you@example.com"
    }
  }
}
```

### 1.6.7.1.2. 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 1.18. Network parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>networking</strong></td>
<td>The configuration for the cluster network.</td>
<td>Object</td>
</tr>
<tr>
<td><strong>networking.network</strong></td>
<td>The cluster network provider Container Network Interface (CNI) plug-in to install.</td>
<td>Either OpenShiftSDN or OVNKubernetes. The default value is OpenShiftSDN.</td>
</tr>
<tr>
<td><strong>networking.clusterNetwork</strong></td>
<td>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.</td>
<td>An array of objects. For example: networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23</td>
</tr>
</tbody>
</table>

**NOTE**

You cannot modify parameters specified by the `networking` object after installation.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>networking.clusterNetwork.hostPrefix</strong></td>
<td>The subnet prefix length to assign to each individual node. For example, if <code>hostPrefix</code> is set to 23 then each node is assigned a /23 subnet out of the given <code>cidr</code>. A <code>hostPrefix</code> value of 23 provides 510 (2^(32 - 23) - 2) pod IP addresses.</td>
<td>A subnet prefix. The default value is 23.</td>
</tr>
<tr>
<td><strong>networking.serviceNetwork</strong></td>
<td>The IP address block for services. The default value is 172.30.0.0/16.</td>
<td>An array with an IP address block in CIDR format. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>networking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>serviceNetwork:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 172.30.0.0/16</td>
</tr>
<tr>
<td><strong>networking.machineNetwork</strong></td>
<td>The IP address blocks for machines. If you specify multiple IP address blocks, the blocks must not overlap.</td>
<td>An array of objects. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>networking:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>machineNetwork:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- cidr: 10.0.0.0/16</td>
</tr>
<tr>
<td><strong>networking.machineNetwork.cidr</strong></td>
<td>Required if you use <code>networking.machineNetwork</code>. 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.</td>
<td>An IP network block in CIDR notation. For example, 10.0.0.0/16.</td>
</tr>
</tbody>
</table>

**NOTE**

Set the `networking.machineNetwork` to match the CIDR that the preferred NIC resides in.

---

1.6.7.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 1.19. Optional parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>additionalTrustBundle</td>
<td>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.</td>
<td>String</td>
</tr>
<tr>
<td>compute</td>
<td>The configuration for the machines that comprise the compute nodes.</td>
<td>Array of machine-pool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>compute.architecture</td>
<td>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 <strong>amd64</strong> (the default).</td>
<td>String</td>
</tr>
<tr>
<td>compute.hyperthreading</td>
<td>Whether to enable or disable simultaneous multithreading, or <strong>hyperthreading</strong>, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.</td>
<td><strong>Enabled</strong> or <strong>Disabled</strong></td>
</tr>
<tr>
<td></td>
<td><strong>IMPORTANT</strong> If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</td>
<td></td>
</tr>
<tr>
<td>compute.name</td>
<td>Required if you use <code>compute</code>. The name of the machine pool.</td>
<td>worker</td>
</tr>
<tr>
<td>compute.platform</td>
<td>Required if you use <code>compute</code>. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the <code>controlPlane.platform</code> parameter value.</td>
<td>aws, azure, gcp, openstack, ovirt, vsphere, or{}</td>
</tr>
<tr>
<td>compute.replicas</td>
<td>The number of compute machines, which are also known as worker machines, to provision.</td>
<td>A positive integer greater than or equal to 2. The default value is 3.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>controlPlane</td>
<td>The configuration for the machines that comprise the control plane.</td>
<td>Array of MachinePool objects. For details, see the following &quot;Machine-pool&quot; table.</td>
</tr>
<tr>
<td>controlPlane.architecture</td>
<td>Determines the instruction set architecture of the machines in the pool.</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Currently, heterogeneous clusters are not supported, so all pools must</td>
<td></td>
</tr>
<tr>
<td></td>
<td>specify the same architecture. Valid values are amd64 (the default).</td>
<td></td>
</tr>
<tr>
<td>controlPlane.hypertreading</td>
<td>Whether to enable or disable simultaneous multithreading, or hyperthreading,</td>
<td>Enabled or Disabled</td>
</tr>
<tr>
<td></td>
<td>on control plane machines. By default, simultaneous multithreading is</td>
<td></td>
</tr>
<tr>
<td></td>
<td>enabled to increase the performance of your machines' cores.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>IMPORTANT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you disable simultaneous multithreading, ensure that your capacity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>planning accounts for the dramatically decreased machine performance.</td>
<td></td>
</tr>
<tr>
<td>controlPlane.name</td>
<td>Required if you use controlPlane. The name of the machine pool.</td>
<td>master</td>
</tr>
<tr>
<td>controlPlane.platform</td>
<td>Required if you use controlPlane. Use this parameter to specify the cloud</td>
<td>aws, azure, gcp, openstack, ovirt, vsphere, or{}</td>
</tr>
<tr>
<td></td>
<td>provider that hosts the control plane machines. This parameter value must</td>
<td></td>
</tr>
<tr>
<td></td>
<td>match the compute.platform parameter value.</td>
<td></td>
</tr>
<tr>
<td>controlPlane.replicas</td>
<td>The number of control plane machines to provision.</td>
<td>The only supported value is 3, which is the default value.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>fips</td>
<td>Enable or disable FIPS mode. The default is <code>false</code> (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.</td>
<td>false or true</td>
</tr>
<tr>
<td>imageContentSources</td>
<td>Sources and repositories for the release-image content.</td>
<td>Array of objects. Includes a <code>source</code> and, optionally, <code>mirrors</code>, as described in the following rows of this table.</td>
</tr>
<tr>
<td>imageContentSources.source</td>
<td>Required if you use <code>imageContentSources</code>. Specify the repository that users refer to, for example, in image pull specifications.</td>
<td>String</td>
</tr>
<tr>
<td>imageContentSources.mirrors</td>
<td>Specify one or more repositories that may also contain the same images.</td>
<td>Array of strings</td>
</tr>
<tr>
<td>publish</td>
<td>How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.</td>
<td>Internal or External. To deploy a private cluster, which cannot be accessed from the internet, set <code>publish</code> to Internal. The default value is External.</td>
</tr>
</tbody>
</table>
### 1.6.7.1.4. Additional Google Cloud Platform (GCP) configuration parameters

Additional GCP configuration parameters are described in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform.gcp.networ...</td>
<td>The name of the existing VPC that you want to deploy your cluster to.</td>
<td>String</td>
</tr>
<tr>
<td>platform.gcp.type</td>
<td>The GCP machine type.</td>
<td>The GCP machine type.</td>
</tr>
<tr>
<td>platform.gcp.zones</td>
<td>The availability zones where the installation program creates machines for the specified MachinePool.</td>
<td>A list of valid GCP availability zones, such as us-central1-a, in a YAML sequence.</td>
</tr>
<tr>
<td>platform.gcp.control PlaneSubnet</td>
<td>The name of the existing subnet in your VPC that you want to deploy your control plane machines to.</td>
<td>The subnet name.</td>
</tr>
<tr>
<td>platform.gcp.comput...</td>
<td>The name of the existing subnet in your VPC that you want to deploy your compute machines to.</td>
<td>The subnet name.</td>
</tr>
</tbody>
</table>

### 1.6.7.2. 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.

**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. For example, 

```
sshKey: ssh-ed25519 AAAA...
```

**IMPORTANT**

OpenShift Container Platform 4.5 Installing on GCP
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.

```yaml
apiVersion: v1
baseDomain: example.com
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
        - us-central1-a
        - us-central1-c
  replicas: 3
  compute:
    - hyperthreading: Enabled
    name: worker
    platform:
      gcp:
        type: n2-standard-4
        zones:
          - us-central1-a
          - us-central1-c
        replicas: 3
    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
        region: us-central1
        network: existing_vpc
        controlPlaneSubnet: control_plane_subnet
        computeSubnet: compute_subnet
        pullSecret: '{"auths": ...}'
        fips: false
        sshKey: ssh-ed25519 AAAA...
        publish: Internal
```

1. 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
3. 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. 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.

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.

1.6.7.3. 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:

```yaml
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: http://<username>:<pswd>@<ip>:<port>
  noProxy: example.com
additionalTrustBundle:
  -----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.
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`.

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

1.6.8. 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> \ 
   --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.
1.6.9. 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.

#### 1.6.9.1. 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>
```
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>
```

**1.6.9.3. 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>
```

**1.6.10. 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
   ```

   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
   ```

1.6.1. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting.

1.7. 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.

1.7.1. 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.

### 1.7.2. 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.

### 1.7.3. Configuring your GCP project

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

#### 1.7.3.1. 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.

#### 1.7.3.2. 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 1.21. Required API services

<table>
<thead>
<tr>
<th>API service</th>
<th>Console service name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Deployment Manager V2 API</td>
<td>deploymentmanager.googleapis.com</td>
</tr>
<tr>
<td>Compute Engine API</td>
<td>compute.googleapis.com</td>
</tr>
<tr>
<td>Google Cloud APIs</td>
<td>cloudapis.googleapis.com</td>
</tr>
<tr>
<td>Cloud Resource Manager API</td>
<td>cloudreresourcemanager.googleapis.com</td>
</tr>
<tr>
<td>Google DNS API</td>
<td>dns.googleapis.com</td>
</tr>
<tr>
<td>IAM Service Account Credentials API</td>
<td>iamcredentials.googleapis.com</td>
</tr>
<tr>
<td>Identity and Access Management (IAM) API</td>
<td>iam.googleapis.com</td>
</tr>
<tr>
<td>Service Management API</td>
<td>servicemanagement.googleapis.com</td>
</tr>
<tr>
<td>Service Usage API</td>
<td>serviceusage.googleapis.com</td>
</tr>
<tr>
<td>Google Cloud Storage JSON API</td>
<td>storage-api.googleapis.com</td>
</tr>
<tr>
<td>Cloud Storage</td>
<td>storage-component.googleapis.com</td>
</tr>
</tbody>
</table>

1.7.3.3. 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.

1.7.3.4. 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 1.22. GCP resources used in a default cluster

<table>
<thead>
<tr>
<th>Service</th>
<th>Component</th>
<th>Location</th>
<th>Total resources required</th>
<th>Resources removed after bootstrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service account</td>
<td>IAM</td>
<td>Global</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Firewall rules</td>
<td>Networking</td>
<td>Global</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Health checks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Images</td>
<td>Compute</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Networks</td>
<td>Networking</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routers</td>
<td>Networking</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routes</td>
<td>Networking</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
### 1.7.3.5. 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.

1.7.3.5.1. 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:
The roles are applied to the service accounts that the control plane and compute machines use:

### Table 1.23. GCP service account permissions

<table>
<thead>
<tr>
<th>Account</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane</td>
<td><code>roles/compute.instanceAdmin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/compute.networkAdmin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/compute.securityAdmin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/storage.admin</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/iam.serviceAccountUser</code></td>
</tr>
<tr>
<td>Compute</td>
<td><code>roles/compute.viewer</code></td>
</tr>
<tr>
<td></td>
<td><code>roles/storage.admin</code></td>
</tr>
</tbody>
</table>

#### 1.7.3.6. 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)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
1.7.3.7. 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.

1.7.4. 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.

1.7.4.1. 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:

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

   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:

   ```yaml
   compute:
   - hyperthreading: Enabled
   ```
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.

### 1.7.4.2. 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:

```yaml
name: worker
platform: {}
replicas: 0

apiVersion: v1
baseDomain: my.domain.com

proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: http://<username>:<pswd>@<ip>:<port>
  noProxy: example.com
  additionalTrustBundle: |
      -----BEGIN 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.

### 1.7.4.3. 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>
   ```

   **Example output**

   ```
   INFO Consuming Install Config from target directory
   WARNING Making control-plane schedulable by setting MastersSchedulable to true for
   Scheduler cluster settings
   ```

   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:

```yaml
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. 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:

```bash
$ ./openshift-install create ignition-configs --dir=<installation_directory>
```

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

### 1.7.5. Exporting common variables

#### 1.7.5.1. 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:

  ```shell
  $ jq -r .infraID <installation_directory>/metadata.json
  ```

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

**Example output**

```
openshift-vw9j6
```

The output of this command is your cluster name and a random string.

1.7.5.2. 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
$ 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`
```

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

1.7.6. 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}'
```

1 For `<installation_directory>`, specify the path to the directory that you stored the installation files in.
infra_id is the INFRA_ID infrastructure name from the extraction step.  
region is the region to deploy the cluster into, for example us-central1.  
master_subnet_cidr is the CIDR for the master subnet, for example 10.0.0.0/19.  
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
```

1.7.6.1. 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 1.1. 01_vpc.py Deployment Manager template**

```python
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'],
```

---

OpenShift Container Platform 4.5 Installing on GCP

---

116
1.7.7. 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 1.24. All machines to all machines

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP</td>
<td>N/A</td>
<td>Network reachability tests</td>
</tr>
<tr>
<td>TCP</td>
<td>1936</td>
<td>Metrics</td>
</tr>
<tr>
<td>9000-9999</td>
<td></td>
<td>Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099.</td>
</tr>
<tr>
<td>10250-10259</td>
<td></td>
<td>The default ports that Kubernetes reserves</td>
</tr>
<tr>
<td>10256</td>
<td></td>
<td>openshift-sdn</td>
</tr>
<tr>
<td>UDP</td>
<td>4789</td>
<td>VXLAN and Geneve</td>
</tr>
<tr>
<td>6081</td>
<td></td>
<td>VXLAN and Geneve</td>
</tr>
<tr>
<td>Protocol</td>
<td>Port</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>Kubernetes node port</td>
</tr>
<tr>
<td>Host level services, including the node exporter on ports 9100-9101.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.25. All machines to control plane

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>6443</td>
<td>Kubernetes API</td>
</tr>
</tbody>
</table>

Table 1.26. Control plane machines to control plane machines

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>2379-2380</td>
<td>etcd server and peer ports</td>
</tr>
</tbody>
</table>

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 1.27. API load balancer
### Port 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.

### Port 22623
Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.

**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 1.28. Application Ingress load balancer**

<table>
<thead>
<tr>
<th>Port</th>
<th>Back-end machines (pool members)</th>
<th>Internal</th>
<th>External</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>443</strong></td>
<td>The machines that run the Ingress router pods, compute, or worker, by default.</td>
<td>X</td>
<td>X</td>
<td>HTTPS traffic</td>
</tr>
<tr>
<td><strong>80</strong></td>
<td>The machines that run the Ingress router pods, compute, or worker, by default.</td>
<td>X</td>
<td>X</td>
<td>HTTP traffic</td>
</tr>
</tbody>
</table>
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.

1.7.8. 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:

   ```bash
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. Export the control plane subnet location:

   ```bash
   $ 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:

   ```bash
   ```
Create a `02_infra.yaml` resource definition file:

```
$ cat <<EOF >02_infra.yaml
  imports:
  - path: 02_lb_ext.py
  - path: 02_lb_int.py
  resources:
  - name: cluster-lb-ext
    type: 02_lb_ext.py
    properties:
      infra_id: '${INFRA_ID}'
      region: '${REGION}'
  - name: cluster-lb-int
    type: 02_lb_int.py
    properties:
      cluster_network: '${CLUSTER_NETWORK}'
      control_subnet: '${CONTROL_SUBNET}'
      infra_id: '${INFRA_ID}'
      region: '${REGION}'
      zones:
      - '${ZONE_0}'
      - '${ZONE_1}'
      - '${ZONE_2}'
EOF
```

1. Required only when deploying an external cluster.
2. `infra_id` is the `INFRA_ID` infrastructure name from the extraction step.
3. `region` is the region to deploy the cluster into, for example `us-central1`.
4. `control_subnet` is the URI to the control subnet.
5. `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 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`)
```
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')
```

### 1.7.8.1. 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 1.2. 02_lb_ext.py Deployment Manager template**

```python
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}
```
1.7.8.2. 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 1.3. 02_lb_int.py Deployment Manager template

def GenerateConfig(context):
    
    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': f'$(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']
        }
    }, {
        #'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': [f'$(ref. + context.properties[infra_id] + api-internal-health-check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': f'$(ref. + context.properties[infra_id] + api-internal-backend-service.selfLink)',
            'IPAddress': f'$(ref. + context.properties[infra_id] + cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
        }
    }]

# Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver

You will need this template in addition to the `02_lb_ext.py` template when you create an external cluster.

### 1.7.9. 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**

```python
'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}
```
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: '${INFRA_ID}'
     cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}'
     cluster_network: '${CLUSTER_NETWORK}'

   EOF
   ```

   - `infra_id` is the INFRA_ID infrastructure name from the extraction step.
   - `cluster_domain` is the domain for the cluster, for example openshift.example.com.
   - `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)
   ```
1.7.9.1. 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 1.4. 02_dns.py Deployment Manager template

```python
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}
```

1.7.10. 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:

```yaml
$ 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'
    infra_id: '${INFRA_ID}'
    cluster_network: '${CLUSTER_NETWORK}'
    network_cidr: '${NETWORK_CIDR}'
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:

```bash
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config 03_firewall.yaml
```

### 1.7.10.1. 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:

```python
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}
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}'
   EOF
   ```

   *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:

```bash
$ 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:

```bash
$ 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_SERVICEACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member "serviceAccount:${WORKER_SERVICEACCOUNT}" --role "roles/storage.admin"
```

8. Create a service account key and store it locally for later use:

```bash
$ gcloud iam service-accounts keys create service-account-key.json --iam-account=${MASTER_SERVICE_ACCOUNT}
```

1.7.11.1. 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 1.6. 03_iams.py Deployment Manager template**

```python
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'  
         }
```
1.7.12. 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}"

1.7.13. 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:

   ```bash
   $ 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:

   ```bash
   $ 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:

   ```bash
   $ 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:

   ```bash
   $ cat <<EOF >04_bootstrap.yaml
   imports:
   - path: 04_bootstrap.py

   resources:
   - name: cluster-bootstrap
     type: 04_bootstrap.py
     properties:
       infra_id: '${INFRA_ID}'
   ```
infra_id is the INFRA_ID infrastructure name from the extraction step.

region is the region to deploy the cluster into, for example us-central1.

zone is the zone to deploy the bootstrap instance into, for example us-central1-b.

cluster_network is the selfLink URL to the cluster network.

control_subnet is the selfLink URL to the control subnet.

image is the selfLink URL to the RHCOS image.

machine_type is the machine type of the instance, for example n1-standard-4.

root_volume_size is the boot disk size for the bootstrap machine.

bootstrap_ign is the URL output when creating a signed URL.

6. Create the deployment by using the gcloud CLI:

```bash
$ 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:

```bash
$ 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:

```bash
$ 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}
```

1.7.13.1. 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 1.7. 04_bootstrap.py Deployment Manager template**

```python
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":{}}},"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'}
            ]
        }
    }
```
1.7.14. 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.

```
return {'resources': resources}
```

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:
```
infra_id is the INFRA_ID infrastructure name from the extraction step.

zones are the zones to deploy the control plane instances into, for example us-central1-a, us-central1-b, and us-central1-c.

control_subnet is the selfLink URL to the control subnet.

image is the selfLink URL to the RHCOS image.

machine_type is the machine type of the instance, for example n1-standard-4.

service_account_email is the email address for the master service account that you created.

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:

```bash
$ 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
```

1.7.14.1. 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:

```python
Example 1.8. 05_control_plane.py Deployment Manager template

def GenerateConfig(context):
    resources = [
        
        '{name': context.properties['infra_id'] + '-m-0',
        'type': 'compute.v1.instance',
        'properties': {
            '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': [],
  'serviceAccounts': [],
  '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': []}
1.7.15. 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> \  
   --log-level info
   ```

   
   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
```

1.7.16. 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:

   ```bash
   $ 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:

```bash
$ 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:

```bash
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```bash
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0'
type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}'
    zone: '${ZONE_0}'
    compute_subnet: '${COMPUTE_SUBNET}'
    image: '${CLUSTER_IMAGE}'
    machine_type: 'n1-standard-4'
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}'
    ignition: '${WORKER_IGNITION}'

- name: 'worker-1'
type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}'
    zone: '${ZONE_1}'
    compute_subnet: '${COMPUTE_SUBNET}'
    image: '${CLUSTER_IMAGE}'
    machine_type: 'n1-standard-4'
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}'
    ignition: '${WORKER_IGNITION}'

EOF
```

- **name** is the name of the worker machine, for example **worker-0**.
- **infra_id** is the **INFRA_ID** infrastructure name from the extraction step.
- **zone** is the zone to deploy the worker machine into, for example **us-central1-a**.
- **compute_subnet** is the selfLink URL to the compute subnet.
- **image** is the selfLink URL to the RHCOs image.
- **machine_type** is the machine type of the instance, for example **n1-standard-4**.
- **service_account_email** is the email address for the worker service account that you created.
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
   ```

1.7.16.1. 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 1.9. 06_worker.py Deployment Manager template

```python
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']
    }
```
1.7.17. 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.

1.7.17.1. 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>
```}

1.7.17.2. 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>
```

### 1.7.17.3. 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>
```

### 1.7.18. 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:

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

   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:

   ```bash
   $ oc whoami
   ```

   **Example output**

   ```
   system:admin
   ```

### 1.7.19. 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:

   ```bash
   $ 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:

   ```bash
   $ oc get csr
   ```

   **Example output**

   ```
   NAME        AGE     REQUESTOR                                                                 CONDITION
   csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
   ```
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>
  ```

  `<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:
OpenShift Container Platform 4.5 Installing on GCP

$ 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
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
master-0
master-1
master-2
worker-0
worker-1

STATUS ROLES AGE VERSION
Ready master 73m v1.20.0
Ready master 73m v1.20.0
Ready master 74m v1.20.0
Ready worker 11m v1.20.0
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 .

1.7.20. 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.

148


• 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:

   ```bash
   $ oc -n openshift-ingress get service router-default
   
   Example output
   
   NAME     TYPE         CLUSTER-IP          EXTERNAL-IP          PORT(S)                   AGE
   
   2. Add the A record to your zones:
   
   • To use A records:
     
     i. Export the variable for the router IP address:

     ```bash
     $ 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:

     ```bash
     $ 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:

     ```bash
     $ 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:

     ```bash
     $ oc get --all-namespaces -o jsonpath="$\{range .items[*]\}{range .status.ingress[*]\}{.host \"\n\}\{end\}\{end\}" routes
     
     Example output
1.7.21. 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:
   
   ```bash
   $ ./openshift-install --dir=<installation_directory> wait-for install-complete
   ```

   **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:
   
   ```bash
   $ oc get clusterversion
   ```

   **Example output**
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     STATUS
authentication                             4.5.4     True        False       False      7m56s     4.5.4: 99% complete
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
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      23m
openshift-controller-manager               4.5.4     True        False       False      17m
openshift-controller-manager               4.5.4     True        False       False      17m
openshift-samples                          4.5.4     True        False       False      17m
operator-lifecycle-manager                 4.5.4     True        False       False      16m
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-2compute.internal 1/1 Running 0 35m
kube-system                                             etcd-member-ip-10-0-3-239.us-east-
```
When the current cluster version is **AVAILABLE**, the installation is complete.

### 1.7.22. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting.

### 1.8. 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.
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.

1.8.1. 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.

1.8.2. 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.

1.8.3. 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.

1.8.3.1. 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.

1.8.3.2. 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 1.29. Required API services

<table>
<thead>
<tr>
<th>API service</th>
<th>Console service name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Deployment Manager V2 API</td>
<td>deploymentmanager.googleapis.com</td>
</tr>
<tr>
<td>Compute Engine API</td>
<td>compute.googleapis.com</td>
</tr>
<tr>
<td>Google Cloud APIs</td>
<td>cloudapis.googleapis.com</td>
</tr>
<tr>
<td>Cloud Resource Manager API</td>
<td>cloudresourcemanager.googleapis.com</td>
</tr>
<tr>
<td>Google DNS API</td>
<td>dns.googleapis.com</td>
</tr>
<tr>
<td>IAM Service Account Credentials API</td>
<td>iamcredentials.googleapis.com</td>
</tr>
<tr>
<td>Identity and Access Management (IAM) API</td>
<td>iam.googleapis.com</td>
</tr>
<tr>
<td>Service Management API</td>
<td>servicemanagement.googleapis.com</td>
</tr>
<tr>
<td>Service Usage API</td>
<td>serviceusage.googleapis.com</td>
</tr>
<tr>
<td>Google Cloud Storage JSON API</td>
<td>storage-api.googleapis.com</td>
</tr>
<tr>
<td>Cloud Storage</td>
<td>storage-component.googleapis.com</td>
</tr>
</tbody>
</table>

1.8.3.3. 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 1.30. GCP resources used in a default cluster

<table>
<thead>
<tr>
<th>Service</th>
<th>Component</th>
<th>Location</th>
<th>Total resources required</th>
<th>Resources removed after bootstrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service account</td>
<td>IAM</td>
<td>Global</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Firewall rules</td>
<td>Networking</td>
<td>Global</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Health checks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Images</td>
<td>Compute</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Networks</td>
<td>Networking</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routers</td>
<td>Networking</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routes</td>
<td>Networking</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Subnetworks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Target pools</td>
<td>Networking</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**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`
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.

1.8.3.4. 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.

1.8.3.4.1. 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 1.31. GCP service account permissions**

<table>
<thead>
<tr>
<th>Account</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane</td>
<td>roles/compute.instanceAdmin</td>
</tr>
<tr>
<td></td>
<td>roles/compute.networkAdmin</td>
</tr>
<tr>
<td></td>
<td>roles/compute.securityAdmin</td>
</tr>
<tr>
<td></td>
<td>roles/storage.admin</td>
</tr>
<tr>
<td></td>
<td>roles/iam.serviceAccountUser</td>
</tr>
<tr>
<td>Compute</td>
<td>roles/compute.viewer</td>
</tr>
<tr>
<td></td>
<td>roles/storage.admin</td>
</tr>
</tbody>
</table>

1.8.3.5. 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)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-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)

1.8.3.6. 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.

1.8.4. 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

1.8.4.1. 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.

1.8.4.2. 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>'
       region: '${REGION}'
       master_subnet_cidr: '${MASTER_SUBNET_CIDR}'
       worker_subnet_cidr: '${WORKER_SUBNET_CIDR}'
   EOF
   ```

   a. `infra_id` is the prefix of the network name.
   
   b. `region` is the region to deploy the cluster into, for example `us-central1`.
   
   c. `master_subnet_cidr` is the CIDR for the master subnet, for example `10.0.0.0/19`.
   
   d. `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:
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.

1.8.4.2.1. 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 1.10. 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']
         }
    ]
```
1.8.5. 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.

1.8.5.1. 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:

```bash
$ 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.

1.8.5.2. 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.

```yaml
apiVersion: v1
baseDomain: example.com
controlPlane:
    hyperthreading: Enabled
    name: master
    platform:
        gcp:
            type: n2-standard-4
            zones:
                - us-central1-a
                - us-central1-c
            replicas: 3
        compute:
            - hyperthreading: Enabled
            name: worker
            platform:
                gcp:
                    type: n2-standard-4
                    zones:
```

OpenShift Container Platform 4.5 Installing on GCP
Specify the public DNS on the host project.

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

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.

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.

Specify the main project where the VM instances reside.

Specify the region that your VPC network is in.

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.
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.

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.

### 1.8.5.3. 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:

```yaml
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: http://<username>:<pswd>@<ip>:<port>
  noProxy: example.com
additionalTrustBundle:
  -----BEGIN 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.

### 1.8.5.4. 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:

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

   **Example output**

   ```bash
   INFO Consuming Install Config from target directory
   WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
   ```

   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:

   ```bash
   $ 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:

   ```bash
   $ 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`.  

   ```bash
   $ OPENSHIFT_COMPONENT=cluster-scheduler-02-config
   ```
c. Save and exit the file.

5. Remove the `privateZone` sections from the `<installation_directory>/manifests/cluster-dns-02-config.yml` DNS configuration file:

```yaml
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
  id: mycluster-100419-private-zone
  status: {}  
1 Remove this section completely.
```

6. Configure the cloud provider for your VPC.

   a. Open the `<installation_directory>/manifests/cloud-provider-config.yaml` file.

   b. Add the `network-project-id` parameter and set its value to the ID of project that hosts the shared VPC network.

   c. Add the `network-name` parameter and set its value to the name of the shared VPC network that hosts the OpenShift Container Platform cluster.

   d. 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:

```yaml
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:

```yaml
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
```

1.8.6. Exporting common variables

1.8.6.1. 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:
For `<installation_directory>`, specify the path to the directory that you stored the installation files in.

**Example output**

```
openshift-vw9j6
```

The output of this command is your cluster name and a random string.

### 1.8.6.2. 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:

   ```bash
   $ jq -r .infraID <installation_directory>/metadata.json
   $ export BASE_DOMAIN='<base_domain>'
   $ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
   $ export NETWORK_CIDR='10.0.0.0/16'
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig
   $ 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`
   ```

   **Supply the values for the host project.**

   **For `<installation_directory>`, specify the path to the directory that you stored the installation files in.**
1.8.7. 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 1.32. All machines to all machines**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP</td>
<td>N/A</td>
<td>Network reachability tests</td>
</tr>
<tr>
<td>TCP</td>
<td>1936</td>
<td>Metrics</td>
</tr>
<tr>
<td></td>
<td>9000-9999</td>
<td>Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099.</td>
</tr>
<tr>
<td></td>
<td>10250-10259</td>
<td>The default ports that Kubernetes reserves</td>
</tr>
<tr>
<td></td>
<td>10256</td>
<td>openshift-sdn</td>
</tr>
<tr>
<td>UDP</td>
<td>4789</td>
<td>VXLAN and Geneve</td>
</tr>
<tr>
<td></td>
<td>6081</td>
<td>VXLAN and Geneve</td>
</tr>
<tr>
<td></td>
<td>9000-9999</td>
<td>Host level services, including the node exporter on ports 9100-9101.</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>Kubernetes node port</td>
</tr>
</tbody>
</table>

**Table 1.33. All machines to control plane**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>6443</td>
<td>Kubernetes API</td>
</tr>
</tbody>
</table>

**Table 1.34. Control plane machines to control plane machines**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>2379-2380</td>
<td>etcd server and peer ports</td>
</tr>
</tbody>
</table>

**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 1.35. API load balancer**

<table>
<thead>
<tr>
<th>Port</th>
<th>Back-end machines (pool members)</th>
<th>Internal</th>
<th>External</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6443</td>
<td>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.</td>
<td>X</td>
<td>X</td>
<td>Kubernetes API server</td>
</tr>
<tr>
<td>22623</td>
<td>Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.</td>
<td>X</td>
<td></td>
<td>Machine config server</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Port</th>
<th>Back-end machines (pool members)</th>
<th>Internal</th>
<th>External</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>443</td>
<td>The machines that run the Ingress router pods, compute, or worker, by default.</td>
<td>X</td>
<td>X</td>
<td>HTTPS traffic</td>
</tr>
<tr>
<td>80</td>
<td>The machines that run the Ingress router pods, compute, or worker, by default.</td>
<td>X</td>
<td>X</td>
<td>HTTP traffic</td>
</tr>
</tbody>
</table>

**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.

### 1.8.8. 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
   resources:
   - name: cluster-lb-ext
     type: 02_lb_ext.py
     properties:
       infra_id: '${INFRA_ID}'
       region: '${REGION}'
   - name: cluster-lb-int
     type: 02_lb_int.py
     properties:
       cluster_network: '${CLUSTER_NETWORK}'
       control_subnet: '${CONTROL_SUBNET}'
       infra_id: '${INFRA_ID}'
   ```
1. Required only when deploying an external cluster.

2. infra_id is the INFRA_ID infrastructure name from the extraction step.

3. region is the region to deploy the cluster into, for example us-central1.

4. control_subnet is the URI to the control subnet.

5. 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`)
   ```

1.8.8.1. 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 1.11. 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': {
               'interval': '30s',
               'path': '/healthz',
               'port': '8080',
               'timeout': '5s',
               'unhealthyThreshold': 3,
               'healthyThreshold': 2,
               'port': 8080
           }
        }
    ]
```
1.8.8.2. 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 1.12. 02_lb_int.py Deployment Manager template

```python
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']
        }
    }, {
        '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}
```

# Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver

```python
    'name': context.properties['infra_id'] + '-api-internal-health-check',
    'type': 'compute.v1.healthCheck',
    'properties': {
```
You will need this template in addition to the 02_lb_ext.py template when you create an external cluster.
1.8.9. 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: '${INFRA_ID}'
         cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}'
         cluster_network: '${CLUSTER_NETWORK}'
   EOF
   ```

   - `infra_id` is the `INFRA_ID` infrastructure name from the extraction step.
   - `cluster_domain` is the domain for the cluster, for example `openshift.example.com`.
   - `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:

   ```bash
   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 ${HOST_PROJECT_ACCOUNT}
   if [ -f transaction.yaml ]; then rm transaction.yaml; fi
   gcloud --account=${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} --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
   gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
   ```

   b. For an external cluster, also add the external DNS entries:

   ```bash
   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} --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
   gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
   ```

1.8.9.1. 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 1.13. 02_dns.py Deployment Manager template**

```python
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}
```
1.8.10. 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'
         infra_id: '${INFRA_ID}'
         cluster_network: '${CLUSTER_NETWORK}'
         network_cidr: '${NETWORK_CIDR}'
   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:
1.8.10.1. 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 1.14. 03_firewall.py Deployment Manager template

```python
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': ['22']
                }],
                'sourceRanges': [context.properties['allowed_external_cidr']],
                'targetTags': [context.properties['infra_id'] + '-master']
            }
        }
    ]
```

$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config 03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
'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': {
'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']
}],
'sourceTags': [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': ['2379-2380']
}]]
1.8.11. 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:
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}
```

1.8.11.1. 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:

```
def GenerateConfig(context):
    resources = [
        
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            ...  # More properties
        }
    ]
```
1.8.12. 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}"
   ```
1.8.13. 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: '${INFRA_ID}'
  region: '${REGION}'
  zone: '${ZONE_0}'
  cluster_network: '${CLUSTER_NETWORK}'
  control_subnet: '${CONTROL_SUBNET}'
  image: '${CLUSTER_IMAGE}'
  machine_type: 'n1-standard-4'
  root_volume_size: '128'
  bootstrap_ign: '${BOOTSTRAP_IGN}'

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**:

   ```bash
   $ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
   04_bootstrap.yaml
   ```

7. Add the bootstrap instance to the internal load balancer instance group:

   ```bash
   $ 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:

   ```bash
   $ 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}
   ```
1.8.13.1. 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 1.16. 04_bootstrap.py Deployment Manager template

```python
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']
                }
            }
        }
    ]
```
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: '${INFRA_ID}'
      zones:
        - '${ZONE_0}'
        - '${ZONE_1}'
        - '${ZONE_2}'
      control_subnet: '${CONTROL_SUBNET}'
      image: '${CLUSTER_IMAGE}'
      machine_type: 'n1-standard-4'
      root_volume_size: '128'
      service_account_email: '${MASTER_SERVICE_ACCOUNT}'
      ignition: '${MASTER_IGNITION}'
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-
For an external cluster, you must also run the following commands to add the control plane machines to the target pools:

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master
$ 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
```

1.8.14.1. 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:

```
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',
            ]
        }
    ]
```
Get the following variables:

```plaintext
zone': context.properties['zones'][0]
```

Get the following variables:

```plaintext
'zone': context.properties['zones'][1]
```

Get the following variables:

```plaintext
'zone': context.properties['zones'][2]
```

Get the following variables:

```plaintext
'zones': context.properties['zones']
```

Get the following variables:

```plaintext
'infra_id': context.properties['infra_id']
```

Get the following variables:

```plaintext
'image': context.properties['image']
```

Get the following variables:

```plaintext
'machine_type': context.properties['machine_type']
```

Get the following variables:

```plaintext
'root_volume_size': context.properties['root_volume_size']
```

Get the following variables:

```plaintext
'diskTypes/pd-ssd'
```

Get the following variables:

```plaintext
'diskTypes/pd-ssd'
```

Get the following variables:

```plaintext
'diskTypes/pd-ssd'
```

Get the following variables:

```plaintext
'https://www.googleapis.com/auth/cloud-platform'
```

Get the following variables:

```plaintext
'ignition'
```

Get the following variables:

```plaintext
'control_subnet'
```

Get the following variables:

```plaintext
'service_account_email'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```

Get the following variables:

```plaintext
'key': 'user-data'
```
1.8.15. 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.
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
   ```

1.8.16. 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:
b. Export the email address for your service account:

```bash
$ 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:

```bash
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. Create a **06_worker.yaml** resource definition file:

```bash
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}'
    zone: '${ZONE_0}'
    compute_subnet: '${COMPUTE_SUBNET}'
    image: '${CLUSTER_IMAGE}'
    machine_type: 'n1-standard-4'
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}'
    ignition: '${WORKER_IGNITION}'

- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}'
    zone: '${ZONE_1}'
    compute_subnet: '${COMPUTE_SUBNET}'
    image: '${CLUSTER_IMAGE}'
    machine_type: 'n1-standard-4'
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}'
    ignition: '${WORKER_IGNITION}'

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.
image is the selfLink URL to the RHCOS image.

machine_type is the machine type of the instance, for example n1-standard-4.

service_account_email is the email address for the worker service account that you created.

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

1.8.16.1. 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 1.18. 06_worker.py Deployment Manager template

```python
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': {
```
1.8.17. 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.

1.8.17.1. 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>
```

1.8.17.2. 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>
   ```

1.8.17.3. 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>
   ```

1.8.18. 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:

   ```bash
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig
   
   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:

   ```bash
   $ oc whoami
   system:admin
   ```

**1.8.19. 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:

   ```bash
   $ oc get nodes
   
   Example output
   
   NAME     STATUS    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:

   ```bash
   $ oc get csr
   ```
Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>REQUESTOR</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>csr-8b2br</td>
<td>15m</td>
<td>system:serviceaccount:openshift-machine-config-operator:node-bootstrapper</td>
<td>Pending</td>
</tr>
<tr>
<td>csr-8vpns</td>
<td>15m</td>
<td>system:serviceaccount:openshift-machine-config-operator:node-bootstrapper</td>
<td>Pending</td>
</tr>
</tbody>
</table>

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>
  ```

  `<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

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>REQUESTOR</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>csr-bfd72</td>
<td>5m26s</td>
<td>system:node:ip-10-0-50-126.us-east-2.compute.internal</td>
<td>Pending</td>
</tr>
<tr>
<td>csr-c57l</td>
<td>5m26s</td>
<td>system:node:ip-10-0-95-157.us-east-2.compute.internal</td>
<td>Pending</td>
</tr>
</tbody>
</table>

...
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>  

  "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

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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>master-0</td>
<td>Ready</td>
<td>master</td>
<td>73m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>master-1</td>
<td>Ready</td>
<td>master</td>
<td>73m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>master-2</td>
<td>Ready</td>
<td>master</td>
<td>74m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>worker-0</td>
<td>Ready</td>
<td>worker</td>
<td>11m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>worker-1</td>
<td>Ready</td>
<td>worker</td>
<td>11m</td>
<td>v1.20.0</td>
</tr>
</tbody>
</table>

   **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](#).

### 1.8.20. 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**
   
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
<th>AGE</th>
</tr>
</thead>
</table>
   
   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 --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
     $ 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}
     
     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} --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
     $ 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{"\n"\n"\n}{end}{end}' routes
```

1.8.21. 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"
```

1.8.21.1. 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.
### 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}
     
     $ 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.
```

### 1.8.22. 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:

   ```bash
  =./openshift-install --dir=<installation_directory> wait-for install-complete
   ```

   **Example output**

   ```
   INFO Waiting up to 30m0s for the cluster to initialize...
   ```

   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:

   ```bash
   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):

   ```bash
   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
   ```
Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

### Example output

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>kube-system</td>
<td>etcd-member-ip-10-0-3-111.us-east-2.compute.internal</td>
</tr>
<tr>
<td>kube-system</td>
<td>etcd-member-ip-10-0-3-239.us-east-2.compute.internal</td>
</tr>
<tr>
<td>kube-system</td>
<td>etcd-member-ip-10-0-3-24.us-east-2.compute.internal</td>
</tr>
<tr>
<td>openshift-apiserver-operator</td>
<td>openshift-apiserver-operator-6d6674f4f4-h72t</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>apiserver-fm48r</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>apiserver-fxkvw</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>apiserver-q85nm</td>
</tr>
<tr>
<td>openshift-service-ca-operator</td>
<td>openshift-service-ca-operator-66ff6dc6cd-9r257</td>
</tr>
<tr>
<td>openshift-service-ca</td>
<td>apiservice-cabundle-injector-695b6bcbc-cl5hm</td>
</tr>
</tbody>
</table>
1.8.23. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting.

1.9. 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-provided 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.

1.9.1. 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.

### 1.9.2. Configuring your GCP project

Before you can install OpenShift Container Platform, you must configure a Google Cloud Platform (GCP) project to host it.

#### 1.9.2.1. 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.

### 1.9.2.2. 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 1.37. Required API services**

<table>
<thead>
<tr>
<th>API service</th>
<th>Console service name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Engine API</td>
<td>compute.googleapis.com</td>
</tr>
</tbody>
</table>
### 1.9.2.3. 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](https://domains.google.com/).

2. Create a public hosted zone for your domain or subdomain in your GCP project. See [Creating public zones](https://cloud.google.com/dns/docs/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](https://cloud.google.com/dns/docs/public-zones) 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.

1.9.2.4. 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 1.38. GCP resources used in a default cluster

<table>
<thead>
<tr>
<th>Service</th>
<th>Component</th>
<th>Location</th>
<th>Total resources required</th>
<th>Resources removed after bootstrap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service account</td>
<td>IAM</td>
<td>Global</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Firewall rules</td>
<td>Networking</td>
<td>Global</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Forwarding rules</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Health checks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Images</td>
<td>Compute</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Networks</td>
<td>Networking</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routers</td>
<td>Networking</td>
<td>Global</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Routes</td>
<td>Networking</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Subnetworks</td>
<td>Compute</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Target pools</td>
<td>Networking</td>
<td>Global</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**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
- europe-north1
- europe-west2
- europe-west3
- europe-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.

1.9.2.5. 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.

1.9.2.5.1. 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 1.39. GCP service account permissions**

<table>
<thead>
<tr>
<th>Account</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane</td>
<td>roles/compute.instanceAdmin</td>
</tr>
</tbody>
</table>
### 1.9.2.6. 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)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-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)
1.9.2.7. 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.

1.9.3. 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.

1.9.3.1. 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:

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

1.9.3.2. 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:

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

   **Example output**

   ```
   INFO Consuming Install Config from target directory
   WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
   ```

   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:

   ```bash
   $ 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:

   ```bash
   $ 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

1.9.4. Exporting common variables
1.9.4.1. 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
```

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

Example output

```
openshift-vw9j6
```

The output of this command is your cluster name and a random string.

1.9.4.2. 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:

```bash
$ 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
$ 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`
```

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

1.9.5. 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:

```bash
$ cat <<EOF >01_vpc.yaml
imports:
  - path: 01_vpc.py

resources:
  - name: cluster-vpc
type: 01_vpc.py
properties:
  infra_id: '${INFRA_ID}'
```

For `<installation_directory>`, specify the path to the directory that you stored the installation files in.
infra_id is the INFRA_ID infrastructure name from the extraction step.

region is the region to deploy the cluster into, for example us-central1.

master_subnet_cidr is the CIDR for the master subnet, for example 10.0.0.0/19.

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:

```bash
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

1.9.5.1. 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 1.19. 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'],
            }
        }
    ]
1.9.6. 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 1.40. All machines to all machines

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP</td>
<td>N/A</td>
<td>Network reachability tests</td>
</tr>
<tr>
<td>TCP</td>
<td>1936</td>
<td>Metrics</td>
</tr>
<tr>
<td></td>
<td>9000-9999</td>
<td>Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099.</td>
</tr>
<tr>
<td></td>
<td>10250-10259</td>
<td>The default ports that Kubernetes reserves</td>
</tr>
<tr>
<td></td>
<td>10256</td>
<td>openshift-sdn</td>
</tr>
<tr>
<td>UDP</td>
<td>4789</td>
<td>VXLAN and Geneve</td>
</tr>
<tr>
<td></td>
<td>6081</td>
<td>VXLAN and Geneve</td>
</tr>
<tr>
<td>Protocol</td>
<td>Port</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>Kubernetes node port</td>
</tr>
<tr>
<td></td>
<td>9000-9999</td>
<td>Host level services, including the node exporter on ports 9100-9101.</td>
</tr>
</tbody>
</table>

**Table 1.41. All machines to control plane**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>6443</td>
<td>Kubernetes API</td>
</tr>
</tbody>
</table>

**Table 1.42. Control plane machines to control plane machines**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>2379-2380</td>
<td>etcd server and peer ports</td>
</tr>
</tbody>
</table>

**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 1.43. API load balancer**

| Port | Back-end machines (pool members) | Internal | External | Description |
|------|----------------------------------|----------|----------|-------------|-------------|

224
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.

### Table 1.44. Application Ingress load balancer

<table>
<thead>
<tr>
<th>Port</th>
<th>Back-end machines (pool members)</th>
<th>Internal</th>
<th>External</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>443</td>
<td>The machines that run the Ingress router pods, compute, or worker, by default.</td>
<td>X</td>
<td>X</td>
<td>HTTPS traffic</td>
</tr>
<tr>
<td>80</td>
<td>The machines that run the Ingress router pods, compute, or worker, by default.</td>
<td>X</td>
<td>X</td>
<td>HTTP traffic</td>
</tr>
</tbody>
</table>

**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:
**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.

### 1.9.7. 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:
      ```bash
      $ export CLUSTER_NETWORK=('gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink')
      ```
   b. Export the control plane subnet location:
      ```bash
      $ 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:
      ```bash
      ```
Create a `02_infra.yaml` resource definition file:

```bash
$ cat <<EOF >02_infra.yaml
imports:
- path: 02lb_ext.py
- path: 02lb_int.py
resources:
- name: cluster-lb-ext
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}'
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: ['${ZONE_0}', '${ZONE_1}', '${ZONE_2}']
EOF
```

1. Required only when deploying an external cluster.
2. `infra_id` is the `INFRA_ID` infrastructure name from the extraction step.
3. `region` is the region to deploy the cluster into, for example `us-central1`.
4. `control_subnet` is the URI to the control subnet.
5. `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:

```bash
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. Export the cluster IP address:

```bash
$ 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`)  
```
For an external cluster, also export the cluster public IP address:

```bash
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

7. For an external cluster, also export the cluster public IP address:

```bash
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

1.9.7.1. 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:

```python
def GenerateConfig(context):

    resources = [
        {'name': context.properties['infra_id'] + '-cluster-public-ip',
         'type': 'compute.v1.address',
         'properties': {
             'region': context.properties['region']
         }},
        {'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}
```

OpenShift Container Platform 4.5 Installing on GCP
1.9.7.2. 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 1.21. 02_lb_int.py Deployment Manager template

```python
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',
```
You will need this template in addition to the 02_lb_ext.py template when you create an external cluster.

1.9.8. 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.

```python
for zone in context.properties['zones']:
    resources.append(
        {'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
         'type': 'compute.v1.instanceGroup',
         'properties': {
             '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.

1.9.8. 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:

```yaml
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
type: 02_dns.py
properties:
  infra_id: '${INFRA_ID}'
  cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}'
  cluster_network: '${CLUSTER_NETWORK}'
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:

```bash
$ 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:
   ```bash
   $ 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
   $ 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}
   ```

   b. For an external cluster, also add the external DNS entries:
   ```bash
   $ 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}
   ```
1.9.8.1. 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 1.22. 02_dns.py Deployment Manager template

```python
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}
```

1.9.9. 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:

```yaml
$ 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'
  infra_id: '${INFRA_ID}'
  cluster_network: '${CLUSTER_NETWORK}'
  network_cidr: '${NETWORK_CIDR}'
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:

```bash
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config 03_firewall.yaml
```

### 1.9.9.1. 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 1.23. 03_firewall.py Deployment Manager template**

```python
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']
}
1.9.10. Creating IAM roles in GCP

```json
'rename': 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}
```
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}'
   EOF
   ```

   *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:

```bash
$ 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:

```bash
$ 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:

```bash
$ gcloud iam service-accounts keys create service-account-key.json --iam-account=${MASTER_SERVICE_ACCOUNT}
```

1.9.10.1. 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 1.24. 03_iam.py Deployment Manager template**

```python
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'
        }
    }
```
1.9.11. 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}"

1.9.12. 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:

   ```sh
   $ 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:

   ```sh
   $ 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:

   ```sh
   $ 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:

   ```sh
   $ cat <<EOF >04_bootstrap.yaml
   imports:
   - path: 04_bootstrap.py

   resources:
   - name: cluster-bootstrap
     type: 04_bootstrap.py
     properties:
       infra_id: '${INFRA_ID}'

   EOF
   ```
infra_id is the INFRA_ID infrastructure name from the extraction step.

region is the region to deploy the cluster into, for example us-central1.

zone is the zone to deploy the bootstrap instance into, for example us-central1-b.

cluster_network is the selfLink URL to the cluster network.

control_subnet is the selfLink URL to the control subnet.

image is the selfLink URL to the RHCOS image.

machine_type is the machine type of the instance, for example n1-standard-4.

root_volume_size is the boot disk size for the bootstrap machine.

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}
```

1.9.12.1. 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:

```
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":{}}},"networkd":{},"passwd":{},"systemd":{}},"timeouts":{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":{}}
            ],
            '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',
```
1.9.13. 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}'
    zones: '
      - ${ZONE_0}'
      - ${ZONE_1}'
      - ${ZONE_2}'
    control_subnet: '${CONTROL_SUBNET}'
    image: '${CLUSTER_IMAGE}'
    machine_type: 'n1-standard-4'
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}'
    ignition: '${MASTER_IGNITION}'

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
```

1.9.13.1. 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 1.26. 05_control_plane.py Deployment Manager template**

```python
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]
}
},

'networkInterfaces': [],
'metadata': [],
'serviceAccounts': [],
'tags': [],
'zone': context.properties['zones'][1]
},

'networkInterfaces': [],
'metadata': [],
'serviceAccounts': [],
'tags': [],
'zone': context.properties['zones'][1]
}
1.9.14. 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:

   ```bash
   $ ./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

1.9.15. 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:

   
   ```bash
   $ 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:

   ```bash
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. Create a **06_worker.yaml** resource definition file:

   ```bash
   $ cat <<EOF >06_worker.yaml
   imports:
   - path: 06_worker.py
   
   resources:
   - name: 'worker-0'
     type: 06_worker.py
     properties:
       infra_id: '${INFRA_ID}'
       zone: '${ZONE_0}'
       compute_subnet: '${COMPUTE_SUBNET}'
       image: '${CLUSTER_IMAGE}'
       machine_type: 'n1-standard-4'
       root_volume_size: '128'
       service_account_email: '${WORKER_SERVICE_ACCOUNT}'
       ignition: '${WORKER_IGNITION}'
   - name: 'worker-1'
     type: 06_worker.py
     properties:
       infra_id: '${INFRA_ID}'
       zone: '${ZONE_1}'
       compute_subnet: '${COMPUTE_SUBNET}'
       image: '${CLUSTER_IMAGE}'
       machine_type: 'n1-standard-4'
       root_volume_size: '128'
       service_account_email: '${WORKER_SERVICE_ACCOUNT}'
       ignition: '${WORKER_IGNITION}'
   EOF
   ```

**name** is the name of the worker machine, for example **worker-0**.

**infra_id** is the **INFRA_ID** infrastructure name from the extraction step.

**zone** is the zone to deploy the worker machine into, for example **us-central1-a**.

**compute_subnet** is the **selfLink** URL to the compute subnet.

**image** is the **selfLink** URL to the RHCOS image.

**machine_type** is the machine type of the instance, for example **n1-standard-4**.

**service_account_email** is the email address for the worker service account that you created.
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:

   ```bash
   $ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config 06_worker.yaml
   ```

1.9.15.1. 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 1.27. 06_worker.py Deployment Manager template**

```python
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']
        }
    }]
1.9.16. 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:

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

   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:

   ```bash
   $ oc whoami
   ```

   Example output

   ```bash
   system:admin
   ```

1.9.17. 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:

   ```bash
   $ oc get nodes
   ```
Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>master-0</td>
<td>Ready</td>
<td>master</td>
<td>63m</td>
<td>v1.18.3</td>
</tr>
<tr>
<td>master-1</td>
<td>Ready</td>
<td>master</td>
<td>63m</td>
<td>v1.18.3</td>
</tr>
<tr>
<td>master-2</td>
<td>Ready</td>
<td>master</td>
<td>64m</td>
<td>v1.18.3</td>
</tr>
<tr>
<td>worker-0</td>
<td>NotReady</td>
<td>worker</td>
<td>76s</td>
<td>v1.18.3</td>
</tr>
<tr>
<td>worker-1</td>
<td>NotReady</td>
<td>worker</td>
<td>70s</td>
<td>v1.18.3</td>
</tr>
</tbody>
</table>

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
```

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
```
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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>REQUESTOR</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>csr-bfd72</td>
<td>5m26s</td>
<td>system:node:ip-10-0-50-126.us-east-2.compute.internal</td>
<td>Pending</td>
</tr>
<tr>
<td>csr-c57lv</td>
<td>5m26s</td>
<td>system:node:ip-10-0-95-157.us-east-2.compute.internal</td>
<td>Pending</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>
```

<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
```

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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>master-0</td>
<td>Ready</td>
<td>master</td>
<td>73m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>master-1</td>
<td>Ready</td>
<td>master</td>
<td>73m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>master-2</td>
<td>Ready</td>
<td>master</td>
<td>74m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>worker-0</td>
<td>Ready</td>
<td>worker</td>
<td>11m</td>
<td>v1.20.0</td>
</tr>
<tr>
<td>worker-1</td>
<td>Ready</td>
<td>worker</td>
<td>11m</td>
<td>v1.20.0</td>
</tr>
</tbody>
</table>

**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](#).
1.9.18. 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:

   ```bash
   $ oc -n openshift-ingress get service router-default
   
   NAME             TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)                      AGE
   
   $ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
   
   $ 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
   ```

2. Add the A record to your zones:

   - To use A records:
     
     i. Export the variable for the router IP address:

     ```bash
     $ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-headers | awk '{print $4}'`
     
     $ 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
     ```

Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
<th>AGE</th>
</tr>
</thead>
</table>
iii. For an external cluster, also add the A record to the public zones:

```bash
$ 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:

```bash
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

**Example output**

```bash
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
```

### 1.9.19. 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:

```bash
$ ./openshift-install --dir=<installation_directory> wait-for install-complete
```

**Example output**

```bash
INFO Waiting up to 30m0s for the cluster to initialize...
```

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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>AVAILABLE</th>
<th>PROGRESSING</th>
<th>SINCE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td></td>
<td>False</td>
<td>True</td>
<td>24m</td>
<td>Working towards 4.5.4: 99% complete</td>
</tr>
</tbody>
</table>

   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**

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>AVAILABLE</th>
<th>PROGRESSING</th>
<th>DEGRADED</th>
<th>SINCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>7m56s</td>
</tr>
<tr>
<td>cloud-credential</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>31m</td>
</tr>
<tr>
<td>cluster-autoscaler</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
<tr>
<td>console</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>10m</td>
</tr>
<tr>
<td>csi-snapshot-controller</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
<tr>
<td>dns</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>22m</td>
</tr>
<tr>
<td>etcd</td>
<td>4.5.4</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>25s</td>
</tr>
<tr>
<td>image-registry</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
<tr>
<td>ingress</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
<tr>
<td>insights</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>17m</td>
</tr>
<tr>
<td>kube-apiserver</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>19m</td>
</tr>
<tr>
<td>kube-controller-manager</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>20m</td>
</tr>
<tr>
<td>kube-scheduler</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>20m</td>
</tr>
<tr>
<td>kube-storage-version-migrator</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
<tr>
<td>machine-api</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>22m</td>
</tr>
<tr>
<td>machine-config</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>22m</td>
</tr>
<tr>
<td>marketplace</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
<tr>
<td>monitoring</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>10m</td>
</tr>
<tr>
<td>network</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>23m</td>
</tr>
<tr>
<td>node-tuning</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>23m</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>17m</td>
</tr>
<tr>
<td>openshift-controller-manager</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>15m</td>
</tr>
<tr>
<td>openshift-samples</td>
<td>4.5.4</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>16m</td>
</tr>
</tbody>
</table>
Run the following command to view your cluster pods:

```
$ oc get pods --all-namespaces
```

Example output

<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>kube-system</td>
<td>etcd-member-ip-10-0-3-111.us-east-2.compute.internal</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
</tr>
<tr>
<td>kube-system</td>
<td>etcd-member-ip-10-0-3-239.us-east-2.compute.internal</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>37m</td>
</tr>
<tr>
<td>kube-system</td>
<td>etcd-member-ip-10-0-3-24.us-east-2.compute.internal</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
</tr>
<tr>
<td>openshift-apiserver-operator</td>
<td>openshift-apiserver-operator-6d6674f4f4-h712t</td>
<td>1/1</td>
<td>Running</td>
<td>1</td>
<td>37m</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>apiserver-fm48r</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>30m</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>apiserver-fxkvv</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
<tr>
<td>openshift-apiserver</td>
<td>apiserver-q85nm</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>29m</td>
</tr>
<tr>
<td>openshift-service-ca-operator</td>
<td>openshift-service-ca-operator-66ff6dc6cd-9r257</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>37m</td>
</tr>
<tr>
<td>openshift-service-ca</td>
<td>apiservice-cabundle-injector-695b6bc1bc-cl5hm</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
</tr>
<tr>
<td>openshift-service-ca</td>
<td>configmap-cabundle-injector-8498544d7-25qn6</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
</tr>
<tr>
<td>openshift-service-ca</td>
<td>service-serving-cert-signer-6445fc9c6-wqdn</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>35m</td>
</tr>
<tr>
<td>openshift-service-catalog-apiserver-operator</td>
<td>openshift-service-catalog-apiserver-operator-549f44668b-b5q2w</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>32m</td>
</tr>
<tr>
<td>openshift-service-catalog-controller-manager-operator</td>
<td>openshift-service-catalog-controller-manager-operator-b78cr2lm</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>31m</td>
</tr>
</tbody>
</table>

When the current cluster version is **AVAILABLE**, the installation is complete.

### 1.9.20. 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.

1.10. UNINSTALLING A CLUSTER ON GCP

You can remove a cluster that you deployed to Google Cloud Platform (GCP).

1.10.1. 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:

   ```bash
   $ ./openshift-install destroy cluster --dir=<installation_directory> --log-level=info
   ``

   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.