



OpenShift Container Platform 3.11

Installing Clusters

OpenShift Container Platform 3.11 Installing Clusters

OpenShift Container Platform 3.11 Installing Clusters

OpenShift Container Platform 3.11 Installing Clusters

Legal Notice

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

Install your OpenShift Container Platform 3.11 cluster with this guide

Table of Contents

CHAPTER 1. PLANNING YOUR INSTALLATION	6
1.1. INITIAL PLANNING	6
1.2. SIZING CONSIDERATIONS	6
1.3. ENVIRONMENT SCENARIOS	6
1.3.1. Single master and node on one system	7
1.3.2. Single master and multiple nodes	7
1.3.3. Multiple masters using native HA	7
1.3.4. Multiple Masters Using Native HA with External Clustered etcd	7
1.3.5. Stand-alone registry	8
1.4. INSTALLATION TYPES FOR SUPPORTED OPERATING SYSTEMS	8
1.4.1. Required images for system containers	9
1.4.2. systemd service names	9
1.4.3. File path locations	9
1.4.4. Storage requirements	9
CHAPTER 2. SYSTEM AND ENVIRONMENT REQUIREMENTS	11
2.1. SYSTEM REQUIREMENTS	11
2.1.1. Red Hat subscriptions	11
2.1.2. Minimum hardware requirements	11
2.1.3. Production level hardware requirements	13
2.1.4. Red Hat Gluster Storage hardware requirements	13
2.1.5. Monitoring hardware requirements	14
2.1.6. SELinux requirements	14
2.1.7. Optional: Configuring Core Usage	14
2.1.8. Optional: Using OverlayFS	14
2.1.9. Security Warning	15
2.2. ENVIRONMENT REQUIREMENTS	15
2.2.1. DNS Requirements	15
2.2.1.1. Configuring Hosts to Use DNS	16
2.2.1.2. Configuring a DNS Wildcard	17
2.2.2. Network Access Requirements	17
2.2.2.1. NetworkManager	18
2.2.2.2. Configuring firewalld as the firewall	18
2.2.2.3. Required Ports	18
2.2.3. Persistent Storage	21
2.2.4. Cloud Provider Considerations	21
2.2.4.1. Overriding Detected IP Addresses and Host Names	21
2.2.4.2. Post-Installation Configuration for Cloud Providers	22
CHAPTER 3. PREPARING YOUR HOSTS	23
3.1. OPERATING SYSTEM REQUIREMENTS	23
3.2. SETTING PATH	23
3.3. ENSURING HOST ACCESS	23
3.4. SETTING PROXY OVERRIDES	24
3.5. REGISTERING HOSTS	24
3.6. INSTALLING BASE PACKAGES	25
3.7. INSTALLING DOCKER	26
3.8. CONFIGURING DOCKER STORAGE	27
3.8.1. Configuring OverlayFS	27
3.8.2. Configuring thin pool storage	28
3.8.3. Reconfiguring Docker storage	31
3.8.4. Enabling image signature support	31

3.8.5. Managing container logs	32
3.8.6. Viewing available container logs	33
3.8.7. Blocking local volume usage	33
3.9. RED HAT GLUSTER STORAGE SOFTWARE REQUIREMENTS	34
3.10. NEXT STEPS	35
CHAPTER 4. CONFIGURING YOUR INVENTORY FILE	36
4.1. CUSTOMIZING INVENTORY FILES FOR YOUR CLUSTER	36
4.2. CONFIGURING CLUSTER VARIABLES	36
4.3. CONFIGURING DEPLOYMENT TYPE	43
4.4. CONFIGURING HOST VARIABLES	44
4.5. DEFINING NODE GROUPS AND HOST MAPPINGS	45
4.5.1. Node ConfigMaps	45
4.5.2. Node Group Definitions	45
4.5.3. Mapping Hosts to Node Groups	47
4.5.4. Node Host Labels	48
4.5.4.1. Pod Schedulability on Masters	48
4.5.4.2. Pod Schedulability on Nodes	48
4.5.4.3. Configuring Dedicated Infrastructure Nodes	49
4.6. CONFIGURING MASTER API PORT	50
4.7. CONFIGURING CLUSTER PRE-INSTALL CHECKS	50
4.8. CONFIGURING A REGISTRY LOCATION	52
4.9. CONFIGURING A REGISTRY ROUTE	53
4.10. CONFIGURING ROUTER SHARDING	54
4.11. CONFIGURING RED HAT GLUSTER STORAGE PERSISTENT STORAGE	55
4.11.1. Configuring converged mode	55
4.11.2. Configuring independent mode	56
4.12. CONFIGURING AN OPENSIFT CONTAINER REGISTRY	57
4.12.1. Configuring Registry Storage	57
Option A: NFS Host Group	57
Option B: External NFS Host	57
Upgrading or Installing OpenShift Container Platform with NFS	58
Option C: OpenStack Platform	58
Option D: AWS or Another S3 Storage Solution	58
Option E: converged mode	58
Option F: Google Cloud Storage (GCS) bucket on Google Compute Engine (GCE)	59
Option G: vSphere Volume with vSphere Cloud Provider (VCP)	60
4.13. CONFIGURING GLOBAL PROXY OPTIONS	60
4.14. CONFIGURING THE FIREWALL	62
4.15. CONFIGURING SESSION OPTIONS	63
4.16. CONFIGURING CUSTOM CERTIFICATES	64
4.17. CONFIGURING CERTIFICATE VALIDITY	65
4.18. CONFIGURING CLUSTER MONITORING	65
4.19. CONFIGURING CLUSTER METRICS	65
4.19.1. Configuring Metrics Storage	66
Option A: Dynamic	66
Option B: NFS Host Group	66
Option C: External NFS Host	67
Upgrading or Installing OpenShift Container Platform with NFS	67
4.20. CONFIGURING CLUSTER LOGGING	67
4.20.1. Configuring Logging Storage	68
Option A: Dynamic	68
Option B: NFS Host Group	68

Option C: External NFS Host	69
Upgrading or Installing OpenShift Container Platform with NFS	69
4.21. CUSTOMIZING SERVICE CATALOG OPTIONS	69
4.21.1. Configuring the OpenShift Ansible Broker	70
4.21.1.1. Configuring Persistent Storage for the OpenShift Ansible Broker	70
4.21.1.2. Configuring the OpenShift Ansible Broker for Local APB Development	71
4.21.2. Configuring the Template Service Broker	72
4.22. CONFIGURING WEB CONSOLE CUSTOMIZATION	72
4.23. CONFIGURING THE CLUSTER CONSOLE	74
4.24. CONFIGURING THE OPERATOR LIFECYCLE MANAGER	74
CHAPTER 5. EXAMPLE INVENTORY FILES	76
5.1. OVERVIEW	76
5.2. SINGLE MASTER EXAMPLES	76
5.2.1. Single Master, Single etcd, and Multiple Nodes	76
5.2.2. Single Master, Multiple etcd, and Multiple Nodes	77
5.3. MULTIPLE MASTERS EXAMPLES	79
5.3.1. Multiple Masters Using Native HA with External Clustered etcd	80
5.3.2. Multiple Masters Using Native HA with Co-located Clustered etcd	82
CHAPTER 6. INSTALLING OPENSIFT CONTAINER PLATFORM	85
6.1. PREREQUISITES	85
6.1.1. Running the RPM-based installer	85
6.1.2. Running the containerized installer	86
6.1.2.1. Running the installer as a system container	86
6.1.2.2. Running other playbooks	87
6.1.2.3. Running the installer as a Docker container	87
6.1.2.4. Running the Installation Playbook for OpenStack	89
6.1.3. About the installation playbooks	89
6.2. RETRYING THE INSTALLATION	90
6.3. VERIFYING THE INSTALLATION	92
Verifying Multiple etcd Hosts	92
Verifying Multiple Masters Using HAProxy	93
6.4. OPTIONALLY SECURING BUILDS	93
6.5. KNOWN ISSUES	93
6.6. WHAT'S NEXT?	94
CHAPTER 7. DISCONNECTED INSTALLATION	95
7.1. PREREQUISITES	95
7.2. OBTAINING REQUIRED SOFTWARE PACKAGES AND IMAGES	95
7.2.1. Obtaining OpenShift Container Platform packages	95
7.2.2. Obtaining images	97
7.2.3. Exporting images	100
7.3. PREPARE AND POPULATE THE REPOSITORY SERVER	102
7.4. POPULATE THE REGISTRY	103
7.5. PREPARING CLUSTER HOSTS	104
7.6. INSTALLING OPENSIFT CONTAINER PLATFORM	104
CHAPTER 8. INSTALLING A STAND-ALONE DEPLOYMENT OF OPENSIFT CONTAINER IMAGE REGISTRY	106
8.1. MINIMUM HARDWARE REQUIREMENTS	106
8.2. SUPPORTED SYSTEM TOPOLOGIES	107
8.3. INSTALLING THE OPENSIFT CONTAINER REGISTRY	107

CHAPTER 9. UNINSTALLING OPENSIFT CONTAINER PLATFORM 111

CHAPTER 10. UNINSTALLING A OPENSIFT CONTAINER PLATFORM CLUSTER 112

 10.1. UNINSTALLING NODES 112

CHAPTER 1. PLANNING YOUR INSTALLATION

You install OpenShift Container Platform by running a series of Ansible playbooks. As you prepare to install your cluster, you create an inventory file that represents your environment and OpenShift Container Platform cluster configuration. While familiarity with Ansible might make this process easier, it is not required.

You can read more about Ansible and its basic usage in the [official documentation](#).

1.1. INITIAL PLANNING

Before you install your production OpenShift Container Platform cluster, you need answers to the following questions:

- *How many pods are required in your cluster?* The [Sizing Considerations](#) section provides limits for nodes and pods so you can calculate how large your environment needs to be.
- *How many hosts do you require in the cluster?* The [Environment Scenarios](#) section provides multiple examples of Single Master and Multiple Master configurations.
- *Does you need a [high availability](#) cluster?* High availability configurations improve fault tolerance. In this situation, you might use the [Multiple Masters Using Native HA](#) example to set up your environment.
- *Is [cluster monitoring](#) required?* The monitoring stack requires additional [system resources](#). Note that the monitoring stack is installed by default. See the [cluster monitoring documentation](#) for more information.
- *Do you want to use Red Hat Enterprise Linux (RHEL) or RHEL Atomic Host as the operating system for your cluster nodes?* If you install OpenShift Container Platform on RHEL, you use an RPM-based installation. On RHEL Atomic Host, you use a system container. [Both installation types](#) provide a working OpenShift Container Platform environment.
- *Which identity provider do you use for [authentication](#)?* If you already use a supported identity provider, configure OpenShift Container Platform to use that identity provider during installation.
- *Is my installation supported if I integrate it with other technologies?* See the [OpenShift Container Platform Tested Integrations](#) for a list of tested integrations.

1.2. SIZING CONSIDERATIONS

Determine how many nodes and pods you require for your OpenShift Container Platform cluster. Cluster scalability correlates to the number of pods in a cluster environment. That number influences the other numbers in your setup. See [Cluster Limits](#) for the latest limits for objects in OpenShift Container Platform.

1.3. ENVIRONMENT SCENARIOS

Use these environment scenarios to help plan your OpenShift Container Platform cluster based on your sizing needs.



NOTE

Moving from a single master cluster to multiple masters after installation is not supported.

In all environments, if your etcd hosts are co-located with master hosts, etcd runs as a static pod on the host. If your etcd hosts are not co-located with master hosts, they run etcd as standalone processes.



NOTE

If you use RHEL Atomic Host, you can configure etcd on only master hosts.

1.3.1. Single master and node on one system

You can install OpenShift Container Platform on a single system for only a development environment. You cannot use an *all-in-one environment* as a production environment.

1.3.2. Single master and multiple nodes

The following table describes an example environment for a single [master](#) (with etcd installed on the same host) and two [nodes](#):

Host Name	Infrastructure Component to Install
master.example.com	Master, etcd, and node
node1.example.com	Node
node2.example.com	

1.3.3. Multiple masters using native HA

The following describes an example environment for three [masters](#), one HAProxy load balancer, and two [nodes](#) using the **native** HA method. etcd runs as static pods on the master nodes:

Host Name	Infrastructure Component to Install
master1.example.com	Master (clustered using native HA) and node and clustered etcd
master2.example.com	
master3.example.com	
lb.example.com	HAProxy to load balance API master endpoints
node1.example.com	Node
node2.example.com	

1.3.4. Multiple Masters Using Native HA with External Clustered etcd

The following describes an example environment for three [masters](#), one HAProxy load balancer, three external clustered [etcd](#) hosts, and two [nodes](#) using the **native** HA method:

Host Name	Infrastructure Component to Install
master1.example.com	Master (clustered using native HA) and node
master2.example.com	
master3.example.com	
lb.example.com	HAProxy to load balance API master endpoints
etcd1.example.com	Clustered etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	Node
node2.example.com	

1.3.5. Stand-alone registry

You can also install OpenShift Container Platform to act as a stand-alone registry using the OpenShift Container Platform's integrated registry. See [Installing a Stand-alone Registry](#) for details on this scenario.

1.4. INSTALLATION TYPES FOR SUPPORTED OPERATING SYSTEMS

Starting in OpenShift Container Platform 3.10, if you use RHEL as the underlying OS for a host, the RPM method is used to install OpenShift Container Platform components on that host. If you use RHEL Atomic Host, the system container method is used on that host. Either installation type provides the same functionality for the cluster, but the operating system you use determines how you manage services and host updates.

An RPM installation installs all services through package management and configures services to run in the same user space, while a system container installation installs services using system container images and runs separate services in individual containers.

When using RPMs on RHEL, all services are installed and updated by package management from an outside source. These packages modify a host's existing configuration in the same user space. With system container installations on RHEL Atomic Host, each component of OpenShift Container Platform is shipped as a container, in a self-contained package, that uses the host's kernel to run. Updated, newer containers replace any existing ones on your host.

The following table and sections outline further differences between the installation types:

Table 1.1. Differences between installation types

	Red Hat Enterprise Linux	RHEL Atomic Host
Installation Type	RPM-based	System container
Delivery Mechanism	RPM packages using yum	System container images using docker
Service Management	systemd	docker and systemd units

1.4.1. Required images for system containers

The system container installation type makes use of the following images:

- **openshift3/ose-node**

By default, all of the above images are pulled from the Red Hat Registry at registry.redhat.io.

If you need to use a private registry to pull these images during the installation, you can specify the registry information ahead of time. Set the following Ansible variables in your inventory file, as required:

```
oreg_url='<registry_hostname>/openshift3/ose-${component}:${version}'
openshift_docker_insecure_registries=<registry_hostname>
openshift_docker_blocked_registries=<registry_hostname>
```



NOTE

You can also set the **openshift_docker_insecure_registries** variable to the IP address of the host. **0.0.0.0/0** is not a valid setting.

The default component inherits the image prefix and version from the **oreg_url** value.

The configuration of additional, insecure, and blocked container registries occurs at the beginning of the installation process to ensure that these settings are applied before attempting to pull any of the required images.

1.4.2. systemd service names

The installation process creates relevant **systemd** units which can be used to start, stop, and poll services using normal **systemctl** commands. For system container installations, these unit names match those of an RPM installation.

1.4.3. File path locations

All OpenShift Container Platform configuration files are placed in the same locations during containerized installation as RPM based installations and will survive **os-tree** upgrades.

However, the default image stream and template files are installed at **/etc/origin/examples/** for Atomic Host installations rather than the standard **/usr/share/openshift/examples/** because that directory is read-only on RHEL Atomic Host.

1.4.4. Storage requirements

RHEL Atomic Host installations normally have a very small root file system. However, the etcd, master, and node containers persist data in the **/var/lib/** directory. Ensure that you have enough space on the root file system before installing OpenShift Container Platform. See the [System Requirements](#) section for details.

CHAPTER 2. SYSTEM AND ENVIRONMENT REQUIREMENTS

2.1. SYSTEM REQUIREMENTS



The hosts in your OpenShift Container Platform environment must meet the following hardware specifications and system-level requirements.

2.1.1. Red Hat subscriptions

You must have an active OpenShift Container Platform subscription on your Red Hat account. If you do not, contact your sales representative for more information.

2.1.2. Minimum hardware requirements

The system requirements vary per host type:

Masters	<ul style="list-style-type: none"> • Physical or virtual system or an instance running on a public or private IaaS. • Base OS: Red Hat Enterprise Linux (RHEL) 7.4 or 7.5 with the "Minimal" installation option and the latest packages from the Extras channel, or RHEL Atomic Host 7.4.5 or later. • Minimum 4 vCPU (additional are strongly recommended). • Minimum 16 GB RAM (additional memory is strongly recommended, especially if etcd is co-located on masters). • Minimum 40 GB hard disk space for the file system containing <code>/var/</code>.  • Minimum 1 GB hard disk space for the file system containing <code>/usr/local/bin/</code>. • Minimum 1 GB hard disk space for the file system containing the system's temporary directory.  • Masters with a co-located etcd require a minimum of 4 cores. Two-core systems do not work.
---------	---

Nodes	<ul style="list-style-type: none"> Physical or virtual system, or an instance running on a public or private IaaS. Base OS: RHEL 7.4 or 7.5 with "Minimal" installation option, or RHEL Atomic Host 7.4.5 or later. NetworkManager 1.0 or later. 1 vCPU. Minimum 8 GB RAM. Minimum 15 GB hard disk space for the file system containing <code>/var/</code>. 1 Minimum 1 GB hard disk space for the file system containing <code>/usr/local/bin/</code>. Minimum 1 GB hard disk space for the file system containing the system's temporary directory. 2 An additional minimum 15 GB unallocated space per system running containers for Docker's storage back end; see Configuring Docker Storage. Additional space might be required, depending on the size and number of containers that run on the node.
External etcd Nodes	<ul style="list-style-type: none"> Minimum 20 GB hard disk space for etcd data. See the Hardware Recommendations section of the CoreOS etcd documentation for information how to properly size your etcd nodes. Currently, OpenShift Container Platform stores image, build, and deployment metadata in etcd. You must periodically prune old resources. If you are planning to leverage a large number of these resources, place etcd on machines with large amounts of memory and fast SSD drives.
Ansible controller	The host that you run the Ansible playbook on must have at least 75MiB of free memory per host in the inventory.

1 Meeting the `/var/` file system sizing requirements in RHEL Atomic Host requires making changes to the default configuration. See [Managing Storage with Docker-formatted Containers](#) for instructions on configuring this during or after installation.

2 The system's temporary directory is determined according to the rules defined in the [tempfile](#) module in Python's standard library.



IMPORTANT

OpenShift Container Platform only supports servers with x86_64 architecture.

You must configure storage for each system that runs a container daemon. For containerized installations, you need storage on masters. Also, by default, the web console runs in containers on masters, and masters need storage to run the web console. Containers run on nodes, so nodes always

require storage. The size of storage depends on workload, the number of containers, the size of the running containers, and the containers' storage requirements. You must also configure storage to run containerized etcd.

2.1.3. Production level hardware requirements

Test or sample environments function with the minimum requirements. For production environments, the following recommendations apply:

Master hosts

In a highly available OpenShift Container Platform cluster with external etcd, a master host needs to meet the minimum requirements and have 1 CPU core and 1.5 GB of memory for each 1000 pods. Therefore, the recommended size of a master host in an OpenShift Container Platform cluster of 2000 pods is the minimum requirements of 2 CPU cores and 16 GB of RAM, plus 2 CPU cores and 3 GB of RAM, totaling 4 CPU cores and 19 GB of RAM.

See [Recommended Practices for OpenShift Container Platform Master Hosts](#) for performance guidance.

Node hosts

The size of a node host depends on the expected size of its workload. As an OpenShift Container Platform cluster administrator, you need to calculate the expected workload and add about 10 percent for overhead. For production environments, allocate enough resources so that a node host failure does not affect your maximum capacity.

For more information, see [Sizing Considerations](#) and [Cluster Limits](#).



IMPORTANT

Oversubscribing the physical resources on a node affects resource guarantees the Kubernetes scheduler makes during pod placement. Learn what measures you can take to [avoid memory swapping](#).

2.1.4. Red Hat Gluster Storage hardware requirements

Any nodes used in a converged mode or independent mode cluster are considered storage nodes. Storage nodes can be grouped into distinct cluster groups, though a single node can not be in multiple groups. For each group of storage nodes:

- A minimum of three storage nodes per group is required.
- Each storage node must have a minimum of 8 GB of RAM. This is to allow running the Red Hat Gluster Storage pods, as well as other applications and the underlying operating system.
 - Each GlusterFS volume also consumes memory on every storage node in its storage cluster, which is about 30 MB. The total amount of RAM should be determined based on how many concurrent volumes are desired or anticipated.
- Each storage node must have at least one raw block device with no present data or metadata. These block devices will be used in their entirety for GlusterFS storage. Make sure the following are not present:
 - Partition tables (GPT or MSDOS)
 - Filesystems or residual filesystem signatures

- LVM2 signatures of former Volume Groups and Logical Volumes
- LVM2 metadata of LVM2 physical volumes

If in doubt, **wipefs -a <device>** should clear any of the above.



IMPORTANT

It is recommended to plan for two clusters: one dedicated to storage for infrastructure applications (such as an OpenShift Container Registry) and one dedicated to storage for general applications. This would require a total of six storage nodes. This recommendation is made to avoid potential impacts on performance in I/O and volume creation.

2.1.5. Monitoring hardware requirements

The monitoring stack imposes additional system resource requirements and is installed by default. See the [computing resources recommendations](#) and the [cluster monitoring documentation](#).

2.1.6. SELinux requirements

Security-Enhanced Linux (SELinux) must be enabled on all of the servers before installing OpenShift Container Platform or the installer will fail. Also, configure **SELINUX=enforcing** and **SELINUXTYPE=targeted** in the **/etc/selinux/config** file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes
are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2.1.7. Optional: Configuring Core Usage

By default, OpenShift Container Platform masters and nodes use all available cores in the system they run on. You can choose the number of cores you want OpenShift Container Platform to use by setting the **GOMAXPROCS** environment variable. See the [Go Language documentation](#) for more information, including how the **GOMAXPROCS** environment variable works.

For example, run the following before starting the server to make OpenShift Container Platform only run on one core:

```
# export GOMAXPROCS=1
```

2.1.8. Optional: Using OverlayFS

OverlayFS is a union file system that allows you to overlay one file system on top of another.

As of Red Hat Enterprise Linux 7.4, you have the option to configure your OpenShift Container Platform environment to use OverlayFS. The **overlay2** graph driver is fully supported in addition to the older **overlay** driver. However, Red Hat recommends using **overlay2** instead of **overlay**, because of its speed and simple implementation.

[Comparing the Overlay Versus Overlay2 Graph Drivers](#) has more information about the **overlay** and **overlay2** drivers.

See the [Overlay Graph Driver](#) section of the Atomic Host documentation for instructions on how to enable the **overlay2** graph driver for the Docker service.

2.1.9. Security Warning

OpenShift Container Platform runs containers on hosts in the cluster, and in some cases, such as build operations and the registry service, it does so using privileged containers. Furthermore, those containers access the hosts' Docker daemon and perform **docker build** and **docker push** operations. As such, cluster administrators must be aware of the inherent security risks associated with performing **docker run** operations on arbitrary images as they effectively have root access. This is particularly relevant for **docker build** operations.

Exposure to harmful containers can be limited by assigning specific builds to nodes so that any exposure is limited to those nodes. To do this, see the [Assigning Builds to Specific Nodes](#) section of the Developer Guide. For cluster administrators, see the [Configuring Global Build Defaults and Overrides](#) topic.

You can also use [security context constraints](#) to control the actions that a pod can perform and what it has the ability to access. For instructions on how to enable images to run with **USER** in the Dockerfile, see [Managing Security Context Constraints](#) (requires a user with **cluster-admin** privileges).

For more information, see these articles:

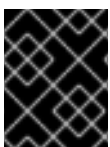
- <http://opensource.com/business/14/7/docker-security-selinux>
- <https://docs.docker.com/engine/security/security/>

2.2. ENVIRONMENT REQUIREMENTS

The following section defines the requirements of the environment containing your OpenShift Container Platform configuration. This includes networking considerations and access to external services, such as Git repository access, storage, and cloud infrastructure providers.

2.2.1. DNS Requirements

OpenShift Container Platform requires a fully functional DNS server in the environment. This is ideally a separate host running DNS software and can provide name resolution to hosts and containers running on the platform.



IMPORTANT

Adding entries into the **/etc/hosts** file on each host is not enough. This file is not copied into containers running on the platform.

Key components of OpenShift Container Platform run themselves inside of containers and use the following process for name resolution:

1. By default, containers receive their DNS configuration file (*/etc/resolv.conf*) from their host.
2. OpenShift Container Platform then sets the pod's first nameserver to the IP address of the node.

As of OpenShift Container Platform 3.2, **dnsmasq** is automatically configured on all masters and nodes. The pods use the nodes as their DNS, and the nodes forward the requests. By default, **dnsmasq** is configured on the nodes to listen on port 53, therefore the nodes cannot run any other type of DNS application.

NOTE

NetworkManager, a program for providing detection and configuration for systems to automatically connect to the network, is required on the nodes in order to populate **dnsmasq** with the DNS IP addresses.

NM_CONTROLLED is set to **yes** by default. If **NM_CONTROLLED** is set to **no**, then the NetworkManager dispatch script does not create the relevant *origin-upstream-dns.conf* dnsmasq file, and you must configure dnsmasq manually.

Similarly, if the **PEERDNS** parameter is set to **no** in the network script, for example, */etc/sysconfig/network-scripts/ifcfg-em1*, then the dnsmasq files are not generated, and the Ansible install will fail. Ensure the **PEERDNS** setting is set to **yes**.

The following is an example set of DNS records:

```
master1    A    10.64.33.100
master2    A    10.64.33.103
node1      A    10.64.33.101
node2      A    10.64.33.102
```

If you do not have a properly functioning DNS environment, you might experience failure with:

- Product installation via the reference Ansible-based scripts
- Deployment of the infrastructure containers (registry, routers)
- Access to the OpenShift Container Platform web console, because it is not accessible via IP address alone

2.2.1.1. Configuring Hosts to Use DNS

Make sure each host in your environment is configured to resolve hostnames from your DNS server. The configuration for hosts' DNS resolution depend on whether DHCP is enabled. If DHCP is:

- Disabled, then configure your network interface to be static, and add DNS nameservers to NetworkManager.
- Enabled, then the NetworkManager dispatch script automatically configures DNS based on the DHCP configuration.

To verify that hosts can be resolved by your DNS server:

1. Check the contents of */etc/resolv.conf*:

```
$ cat /etc/resolv.conf
```

```
# Generated by NetworkManager
search example.com
nameserver 10.64.33.1
# nameserver updated by /etc/NetworkManager/dispatcher.d/99-origin-
dns.sh
```

In this example, 10.64.33.1 is the address of our DNS server.

2. Test that the DNS servers listed in ***/etc/resolv.conf*** are able to resolve host names to the IP addresses of all masters and nodes in your OpenShift Container Platform environment:

```
$ dig <node_hostname> @<IP_address> +short
```

For example:

```
$ dig master.example.com @10.64.33.1 +short
10.64.33.100
$ dig node1.example.com @10.64.33.1 +short
10.64.33.101
```

2.2.1.2. Configuring a DNS Wildcard

Optionally, configure a wildcard for the router to use, so that you do not need to update your DNS configuration when new routes are added.

A wildcard for a DNS zone must ultimately resolve to the IP address of the OpenShift Container Platform [router](#).

For example, create a wildcard DNS entry for **cloudapps** that has a low time-to-live value (TTL) and points to the public IP address of the host where the router will be deployed:

```
*.cloudapps.example.com. 300 IN A 192.168.133.2
```

In almost all cases, when referencing VMs you must use host names, and the host names that you use must match the output of the **hostname -f** command on each node.



WARNING

In your ***/etc/resolv.conf*** file on each node host, ensure that the DNS server that has the wildcard entry is not listed as a nameserver or that the wildcard domain is not listed in the search list. Otherwise, containers managed by OpenShift Container Platform might fail to resolve host names properly.

2.2.2. Network Access Requirements

A shared network must exist between the master and node hosts. If you plan to configure [multiple masters for high-availability](#) using standard cluster installation process, you must also select an IP to be configured as your [virtual IP](#) (VIP) during the installation process. The IP that you select must be routable between all of your nodes, and if you configure using a FQDN it must resolve on all nodes.

2.2.2.1. NetworkManager

NetworkManager, a program for providing detection and configuration for systems to automatically connect to the network, is required on the nodes in order to populate **dnsmasq** with the DNS IP addresses.

NM_CONTROLLED is set to **yes** by default. If **NM_CONTROLLED** is set to **no**, then the NetworkManager dispatch script does not create the relevant **origin-upstream-dns.conf** dnsmasq file, and you must configure dnsmasq manually.

2.2.2.2. Configuring firewalld as the firewall

While iptables is the default firewall, firewalld is recommended for new installations. You can enable firewalld by setting **os_firewall_use_firewalld=true** in [the Ansible inventory file](#).

```
[OSEv3:vars]
os_firewall_use_firewalld=True
```

Setting this variable to **true** opens the required ports and adds rules to the default zone, which ensure that firewalld is configured correctly.



NOTE

Using the firewalld default configuration comes with limited configuration options, and cannot be overridden. For example, while you can set up a storage network with interfaces in multiple zones, the interface that nodes communicate on must be in the default zone.

2.2.2.3. Required Ports

The OpenShift Container Platform installation automatically creates a set of internal firewall rules on each host using [iptables](#). However, if your network configuration uses an external firewall, such as a hardware-based firewall, you must ensure infrastructure components can communicate with each other through specific ports that act as communication endpoints for certain processes or services.

Ensure the following ports required by OpenShift Container Platform are open on your network and configured to allow access between hosts. Some ports are optional depending on your configuration and usage.

Table 2.1. Node to Node

4789	UDP	Required for SDN communication between pods on separate hosts.
-------------	-----	--

Table 2.2. Nodes to Master

53 or 8053	TCP/ UDP	Required for DNS resolution of cluster services (SkyDNS). Installations prior to 3.2 or environments upgraded to 3.2 use port 53. New installations will use 8053 by default so that dnsmasq might be configured.
4789	UDP	Required for SDN communication between pods on separate hosts.

443 or 8443	TCP	Required for node hosts to communicate to the master API, for the node hosts to post back status, to receive tasks, and so on.
-------------	-----	--

Table 2.3. Master to Node

4789	UDP	Required for SDN communication between pods on separate hosts.
10250	TCP	The master proxies to node hosts via the Kubelet for oc commands.
10010	TCP	If using CRI-O, open this port to allow oc exec and oc rsh operations.

Table 2.4. Master to Master

53 or 8053	TCP/ UDP	Required for DNS resolution of cluster services (SkyDNS). Installations prior to 3.2 or environments upgraded to 3.2 use port 53. New installations will use 8053 by default so that dnsmasq might be configured.
2049	TCP/ UDP	Required when provisioning an NFS host as part of the installer.
2379	TCP	Used for standalone etcd (clustered) to accept changes in state.
2380	TCP	etcd requires this port be open between masters for leader election and peering connections when using standalone etcd (clustered).
4789	UDP	Required for SDN communication between pods on separate hosts.

Table 2.5. External to Load Balancer

9000	TCP	If you choose the native HA method, optional to allow access to the HAProxy statistics page.
------	-----	---

Table 2.6. External to Master

443 or 8443	TCP	Required for node hosts to communicate to the master API, for node hosts to post back status, to receive tasks, and so on.
8444	TCP	Port that the controller service listens on. Required to be open for the /metrics and /healthz endpoints.

Table 2.7. IaaS Deployments

22	TCP	Required for SSH by the installer or system administrator.
----	-----	--

53 or 8053	TCP/ UDP	Required for DNS resolution of cluster services (SkyDNS). Installations prior to 3.2 or environments upgraded to 3.2 use port 53. New installations will use 8053 by default so that dnsmasq might be configured. Only required to be internally open on master hosts.
80 or 443	TCP	For HTTP/HTTPS use for the router. Required to be externally open on node hosts, especially on nodes running the router.
1936	TCP	(Optional) Required to be open when running the template router to access statistics. Can be open externally or internally to connections depending on if you want the statistics to be expressed publicly. Can require extra configuration to open. See the Notes section below for more information.
2379 and 2380	TCP	For standalone etcd use. Only required to be internally open on the master host. 2379 is for server-client connections. 2380 is for server-server connections, and is only required if you have clustered etcd.
4789	UDP	For VxLAN use (OpenShift SDN). Required only internally on node hosts.
8443	TCP	For use by the OpenShift Container Platform web console, shared with the API server.
10250	TCP	For use by the Kubelet. Required to be externally open on nodes.

Notes

- In the above examples, port **4789** is used for User Datagram Protocol (UDP).
- When deployments are using the SDN, the pod network is accessed via a service proxy, unless it is accessing the registry from the same node the registry is deployed on.
- OpenShift Container Platform internal DNS cannot be received over SDN. For non-cloud deployments, this will default to the IP address associated with the default route on the master host. For cloud deployments, it will default to the IP address associated with the first internal interface as defined by the cloud metadata.
- The master host uses port **10250** to reach the nodes and does not go over SDN. It depends on the target host of the deployment and uses the computed value of **openshift_public_hostname**.
- Port **1936** can still be inaccessible due to your iptables rules. Use the following to configure iptables to open port **1936**:

```
# iptables -A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp \
--dport 1936 -j ACCEPT
```

Table 2.8. Aggregated Logging

9200	TCP	For Elasticsearch API use. Required to be internally open on any infrastructure nodes so Kibana is able to retrieve logs for display. It can be externally opened for direct access to Elasticsearch by means of a route. The route can be created using oc expose .
9300	TCP	For Elasticsearch inter-cluster use. Required to be internally open on any infrastructure node so the members of the Elasticsearch cluster might communicate with each other.

2.2.3. Persistent Storage

The Kubernetes [persistent volume](#) framework allows you to provision an OpenShift Container Platform cluster with persistent storage using networked storage available in your environment. This can be done after completing the initial OpenShift Container Platform installation depending on your application needs, giving users a way to request those resources without having any knowledge of the underlying infrastructure.

The Configuring Clusters guide provides instructions for cluster administrators on provisioning an OpenShift Container Platform cluster with persistent storage using [NFS](#), [GlusterFS](#), [Ceph RBD](#), [OpenStack Cinder](#), [AWS Elastic Block Store \(EBS\)](#), [GCE Persistent Disks](#), and [iSCSI](#).

2.2.4. Cloud Provider Considerations

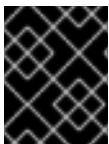
There are certain aspects to take into consideration if installing OpenShift Container Platform on a cloud provider.

- For Amazon Web Services, see the [Permissions](#) and the [Configuring a Security Group](#) sections.
- For OpenStack, see the [Permissions](#) and the [Configuring a Security Group](#) sections.

2.2.4.1. Overriding Detected IP Addresses and Host Names

Some deployments require that the user override the detected host names and IP addresses for the hosts. To see the default values, change to the playbook directory and run the **openshift_facts** playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \
  playbooks/byo/openshift_facts.yml
```



IMPORTANT

For Amazon Web Services, see the [Overriding Detected IP Addresses and Host Names](#) section.

Now, verify the detected common settings. If they are not what you expect them to be, you can override them.

The [Advanced Installation](#) topic discusses the available Ansible variables in greater detail.

Variable	Usage
hostname	<ul style="list-style-type: none">Resolves to the internal IP from the instances themselves.
ip	<ul style="list-style-type: none">The internal IP address of the instance.
public_hostname	<ul style="list-style-type: none">Resolves to the external IP from hosts outside of the cloud.Provider openshift_public_hostname overrides.
public_ip	<ul style="list-style-type: none">The externally accessible IP address associated with the instance.openshift_public_ip overrides.
use_openshift_sdn	<ul style="list-style-type: none">For all clouds but GCE, set to true.openshift_use_openshift_sdn overrides.

2.2.4.2. Post-Installation Configuration for Cloud Providers

Following the installation process, you can configure OpenShift Container Platform for [AWS](#), [OpenStack](#), or [GCE](#).

CHAPTER 3. PREPARING YOUR HOSTS

Before you install OpenShift Container Platform, you must prepare the node hosts. They must meet the following requirements.

3.1. OPERATING SYSTEM REQUIREMENTS

Master and node hosts must use a base installation of Red Hat Enterprise Linux (RHEL) 7.4 or 7.5 with the latest packages from the Extras channel or RHEL Atomic Host 7.4.2 or later. See the following documentation for the respective installation instructions, if required:

- [Red Hat Enterprise Linux 7 Installation Guide](#)
- [Red Hat Enterprise Linux Atomic Host 7 Installation and Configuration Guide](#)

3.2. SETTING PATH

The **PATH** for the root user on each host must contain the following directories:

- **/bin**
- **/sbin**
- **/usr/bin**
- **/usr/sbin**

These directories set by default in a new RHEL 7.x installation.

3.3. ENSURING HOST ACCESS

The OpenShift Container Platform installer requires a user that has access to all hosts. If you want to run the installer as a non-root user, first configure passwordless **sudo** rights each host:

1. Generate an SSH key on the host you run the installation playbook on:

```
# ssh-keygen
```

Do **not** use a password.

2. Distribute the key to the other cluster hosts. You can use a **bash** loop:

```
# for host in master.example.com \
  master.example.com \ 1
  node1.example.com \ 2
  node2.example.com; \ 3
do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
done
```

1 2 3 Provide the host name for each cluster host.

3. Confirm that you can access each host that is listed in the loop through SSH.

3.4. SETTING PROXY OVERRIDES

If the `/etc/environment` file on your nodes contains either an `http_proxy` or `https_proxy` value, you must also set a `no_proxy` value in that file to allow open communication between OpenShift Container Platform components.



NOTE

The `no_proxy` parameter in `/etc/environment` file is not the same value as the global proxy values that you set in your inventory file. The global proxy values configure specific OpenShift Container Platform services with your proxy settings. See [Configuring Global Proxy Options](#) for details.

If the `/etc/environment` file contains proxy values, define the following values in the `no_proxy` parameter of that file on each node:

- Master and node host names or their domain suffix.
- Other internal host names or their domain suffix.
- Etcd IP addresses. You must provide IP addresses and not host names because `etcd` access is controlled by IP address.
- Kubernetes IP address, by default `172.30.0.1`. Must be the value set in the `openshift_portal_net` parameter in your inventory file.
- Kubernetes internal domain suffix, `cluster.local`.
- Kubernetes internal domain suffix, `.svc`.



NOTE

Because `no_proxy` does not support CIDR, you can use domain suffixes.

If you use either an `http_proxy` or `https_proxy` value, your `no_proxy` parameter value resembles the following example:

```
no_proxy=.internal.example.com,10.0.0.1,10.0.0.2,10.0.0.3,.cluster.local,.svc,localhost,127.0.0.1,172.30.0.1
```

3.5. REGISTERING HOSTS

To access the installation packages, you must register each host with Red Hat Subscription Manager (RHSM) and attach an active OpenShift Container Platform subscription.

1. On each host, register with RHSM:

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. Pull the latest subscription data from RHSM:

```
# subscription-manager refresh
```

- 3. List the available subscriptions:

```
# subscription-manager list --available --matches '*OpenShift*'
```

- 4. In the output for the previous command, find the pool ID for an OpenShift Container Platform subscription and attach it:

```
# subscription-manager attach --pool=<pool_id>
```

- 5. Disable all yum repositories:

- a. Disable all the enabled RHSM repositories:

```
# subscription-manager repos --disable="*"
```

- b. List the remaining yum repositories and note their names under **repo id**, if any:

```
# yum repolist
```

- c. Use **yum-config-manager** to disable the remaining yum repositories:

```
# yum-config-manager --disable <repo_id>
```

Alternatively, disable all repositories:

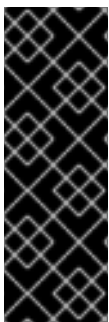
```
yum-config-manager --disable \*
```

Note that this might take a few minutes if you have a large number of available repositories

- 6. Enable only the repositories required by OpenShift Container Platform 3.11:

```
# subscription-manager repos \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ose-3.11-rpms" \
  --enable="rhel-7-server-ansible-2.6-rpms"
```

3.6. INSTALLING BASE PACKAGES



IMPORTANT

If your hosts use RHEL 7.5 and you want to accept OpenShift Container Platform's default **docker** configuration (using OverlayFS storage and all default logging options), do not manually install these packages. These packages are installed when you run the **prerequisites.yml** playbook during [installation](#).

If your hosts use RHEL 7.4 or if they use RHEL 7.5 and you want to customize the **docker** configuration, install these packages.

For RHEL 7 systems:

1. Install the following base packages:

```
# yum install wget git net-tools bind-utils yum-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct
```

2. Update the system to the latest packages:

```
# yum update
# reboot
```

3. Install packages that are required for your installation method:

- If you plan to use the [containerized installer](#), install the following package:

```
# yum install atomic
```

- If you plan to use the [RPM-based installer](#), install the following package:

```
# yum install openshift-ansible
```

This package provides installer utilities and pulls in other packages that the cluster installation process needs, such as Ansible, playbooks, and related configuration files

For RHEL Atomic Host 7 systems:

1. Ensure the host is up to date by upgrading to the latest Atomic tree if one is available:

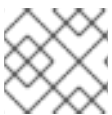
```
# atomic host upgrade
```

2. After the upgrade is completed and prepared for the next boot, reboot the host:

```
# reboot
```

3.7. INSTALLING DOCKER

At this point, install Docker on all master and node hosts. This allows you to configure your [Docker storage options](#) before you install OpenShift Container Platform.



NOTE

The cluster installation process automatically modifies the `/etc/sysconfig/docker` file.

For RHEL 7 systems:

1. Install Docker 1.13:

```
# yum install docker-1.13.1
```

2. Verify that version 1.13 was installed:

```
# rpm -V docker-1.13.1
# docker version
```

For RHEL Atomic Host 7 systems:

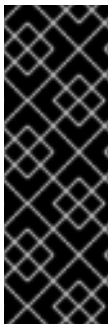
No action is required. Docker is installed, configured, and running by default.

3.8. CONFIGURING DOCKER STORAGE

Containers and the images they are created from are stored in Docker's storage back end. This storage is ephemeral and separate from any [persistent storage](#) allocated to meet the needs of your applications. With *Ephemeral storage*, container-saved data is lost when the container is removed. With *persistent storage*, container-saved data remains if the container is removed.

You must configure storage for all master and node hosts because by default each system runs a container daemon. For containerized installations, you need storage on masters. Also, by default, the web console and etcd, which require storage, run in containers on masters. Containers run on nodes, so storage is always required on them.

The size of storage depends on workload, number of containers, the size of the containers being run, and the containers' storage requirements.

**IMPORTANT**

If your hosts use RHEL 7.5 and you want to accept OpenShift Container Platform's default **docker** configuration (using OverlayFS storage and all default logging options), do not manually install these packages. These packages are installed when you run the **prerequisites.yml** playbook during [installation](#).

If your hosts use RHEL 7.4 or if they use RHEL 7.5 and you want to customize the **docker** configuration, install these packages.

For RHEL 7 systems:

The default storage back end for Docker on RHEL 7 is a thin pool on loopback devices, which is not supported for production use and only appropriate for proof of concept environments. For production environments, you must create a thin pool logical volume and re-configure Docker to use that volume.

Docker stores images and containers in a graph driver, which is a pluggable storage technology, such as **DeviceMapper**, **OverlayFS**, and **Btrfs**. Each has advantages and disadvantages. For example, OverlayFS is faster than DeviceMapper at starting and stopping containers but is not Portable Operating System Interface for Unix (POSIX) compliant because of the architectural limitations of a union file system. See the [Red Hat Enterprise Linux](#) release notes for information on using OverlayFS with your version of RHEL.

For more information about the benefits and limitations of DeviceMapper and OverlayFS, see [Choosing a Graph Driver](#).

For RHEL Atomic Host 7 systems:

The default storage back end for Docker on RHEL Atomic Host is a thin pool logical volume, which is supported for production environments. You must ensure that enough space is allocated for this volume per the Docker storage requirements mentioned in [System Requirements](#).

If you do not have enough space allocated, see [Managing Storage with Docker Formatted Containers](#) for details about using **docker-storage-setup** and basic instructions on storage management in RHEL Atomic Host.

3.8.1. Configuring OverlayFS

OverlayFS is a type of union file system. It allows you to overlay one file system on top of another. Changes are recorded in the upper file system, while the lower file system remains unmodified.

[Comparing the Overlay Versus Overlay2 Graph Drivers](#) has more information about the **overlay** and **overlay2** drivers.

For information about enabling the OverlayFS storage driver for the Docker service, see the [Red Hat Enterprise Linux Atomic Host documentation](#).

3.8.2. Configuring thin pool storage

You can use the **docker-storage-setup** script included with Docker to create a thin pool device and configure Docker's storage driver. You can do this after you install Docker and must do it before you create images or containers. The script reads configuration options from the **/etc/sysconfig/docker-storage-setup** file and supports three options for creating the logical volume:

- Use an additional block device.
- Use an existing, specified volume group.
- Use the remaining free space from the volume group where your root file system is located.

Using an additional block device is the most robust option, but it requires adding another block device to your host before you configure Docker storage. The other options both require leaving free space available when you provision your host. Using the remaining free space in the root file system volume group is known to cause issues with some applications, for example Red Hat Mobile Application Platform (RHMAP).

1. Create the **docker-pool** volume using one of the following three options:

- To use an additional block device:
 - a. **/etc/sysconfig/docker-storage-setup**, set **DEVS** to the path of the block device to use. Set **VG** to the volume group name to create, such as **docker-vg**. For example:

```
# cat <<EOF > /etc/sysconfig/docker-storage-setup
DEVS=/dev/vdc
VG=docker-vg
EOF
```

- b. Run **docker-storage-setup** and review the output to ensure the **docker-pool** volume was created:

```
# docker-storage-setup
[5/1868]
0
Checking that no-one is using this disk right now ...
OK

Disk /dev/vdc: 31207 cylinders, 16 heads, 63 sectors/track
sfdisk: /dev/vdc: unrecognized partition table type

Old situation:
sfdisk: No partitions found

New situation:
```


Units: sectors of 512 bytes, counting from 0

Device	Boot	Start	End	#sectors	Id	System
/dev/vdc1		2048	31457279	31455232	8e	Linux LVM
/dev/vdc2		0	-	0	0	Empty
/dev/vdc3		0	-	0	0	Empty
/dev/vdc4		0	-	0	0	Empty

Warning: partition 1 does not start at a cylinder boundary

Warning: partition 1 does not end at a cylinder boundary

Warning: no primary partition is marked bootable (active)

This does not matter for LILO, but the DOS MBR will not boot this disk.

Successfully wrote the new partition table

Re-reading the partition table ...

If you created or changed a DOS partition, /dev/foo7, say, then use dd(1)

to zero the first 512 bytes: dd if=/dev/zero of=/dev/foo7 bs=512 count=1

(See fdisk(8).)

Physical volume "/dev/vdc1" successfully created

Volume group "docker-vg" successfully created

Rounding up size to full physical extent 16.00 MiB

Logical volume "docker-poolmeta" created.

Logical volume "docker-pool" created.

WARNING: Converting logical volume docker-vg/docker-pool and docker-vg/docker-poolmeta to pool's data and metadata volumes.

THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)

Converted docker-vg/docker-pool to thin pool.

Logical volume "docker-pool" changed.

- To use an existing, specified volume group:

- a. In **/etc/sysconfig/docker-storage-setup**, set **VG** to the volume group. For example:

```
# cat <<EOF > /etc/sysconfig/docker-storage-setup
VG=docker-vg
EOF
```

- b. Then run **docker-storage-setup** and review the output to ensure the **docker-pool** volume was created:

```
# docker-storage-setup
Rounding up size to full physical extent 16.00 MiB
Logical volume "docker-poolmeta" created.
Logical volume "docker-pool" created.
WARNING: Converting logical volume docker-vg/docker-pool and
docker-vg/docker-poolmeta to pool's data and metadata volumes.
THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem
etc.)
Converted docker-vg/docker-pool to thin pool.
Logical volume "docker-pool" changed.
```

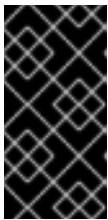
- To use the remaining free space from the volume group where your root file system is located:
 - a. Verify that the volume group where your root file system resides has the required free space, then run **docker-storage-setup** and review the output to ensure the **docker-pool** volume was created:

```
# docker-storage-setup
Rounding up size to full physical extent 32.00 MiB
Logical volume "docker-poolmeta" created.
Logical volume "docker-pool" created.
WARNING: Converting logical volume rhel/docker-pool and
rhel/docker-poolmeta to pool's data and metadata volumes.
THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem
etc.)
Converted rhel/docker-pool to thin pool.
Logical volume "docker-pool" changed.
```

2. Verify your configuration. Confirm that the **/etc/sysconfig/docker-storage** file has **dm.thinpooldev** and **docker-pool** logical volume values:

```
# cat /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS="--storage-driver devicemapper --storage-opt
dm.fs=xfs --storage-opt dm.thinpooldev=/dev/mapper/rhel-docker--pool
--storage-opt dm.use_deferred_removal=true --storage-opt
dm.use_deferred_deletion=true "
```

```
# lvs
LV          VG      Attr              LSize   Pool Origin Data%  Meta%  Move
Log Cpy%Sync Convert
docker-pool rhel    twi-a-t---    9.29g                0.00   0.12
```



IMPORTANT

Before using Docker or OpenShift Container Platform, verify that the **docker-pool** logical volume is large enough to meet your needs. Make the **docker-pool** volume 60% of the available volume group; it will grow to fill the volume group through LVM monitoring.

3. Start or restart Docker.
 - If Docker has never run on the host, enable and start the service, then verify that it is running:


```
# systemctl enable docker
# systemctl start docker
# systemctl is-active docker
```
 - If Docker is already running:
 - a. Re-initialize Docker:

**WARNING**

This will destroy any containers or images currently on the host.

```
# systemctl stop docker
# rm -rf /var/lib/docker/*
# systemctl restart docker
```

- b. Delete any content in the **`/var/lib/docker/`** folder.

3.8.3. Reconfiguring Docker storage

If you need to reconfigure Docker storage after you create the **docker-pool**:

1. Remove the **docker-pool** logical volume.
2. If you use a dedicated volume group, remove the volume group and any associated physical volumes
3. Run **docker-storage-setup** again.

See [Logical Volume Manager Administration](#) for more detailed information about LVM management.

3.8.4. Enabling image signature support

OpenShift Container Platform is capable of cryptographically verifying that images are from trusted sources. The [Container Security Guide](#) provides a high-level description of how image signing works.

You can configure image signature verification using the **atomic** command line interface (CLI), version 1.12.5 or greater. The **atomic** CLI is pre-installed on RHEL Atomic Host systems.

**NOTE**

For more on the **atomic** CLI, see the [Atomic CLI documentation](#).

The following files and directories comprise the trust configuration of a host:

- **`/etc/containers/registries.d/*`**
- **`/etc/containers/policy.json`**

You can manage trust configuration directly on each node or manage the files on a separate host distribute them to the appropriate nodes using Ansible, for example. See the [Container Image Signing Integration Guide](#) for an example of automating file distribution with Ansible.

1. Install the **atomic** package if it is not installed on the host system:

```
$ yum install atomic
```

2. View the current trust configuration:

```
$ atomic trust show
* (default)                                accept
```

The default configuration is to whitelist all registries, which means that no signature verification is configured.

3. Customize your trust configuration. In the following example, you whitelist one registry or namespace, blacklist (reject) untrusted registries, and require signature verification on a vendor registry:

```
$ atomic trust add --type insecureAcceptAnything 172.30.1.1:5000

$ atomic trust add --sigstoretype atomic \
  --pubkeys pub@example.com \
  172.30.1.1:5000/production

$ atomic trust add --sigstoretype atomic \
  --pubkeys /etc/pki/example.com.pub \
  172.30.1.1:5000/production

$ atomic trust add --sigstoretype web \
  --sigstore
https://access.redhat.com/webassets/docker/content/sigstore \
  --pubkeys /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release \
  registry.redhat.io

# atomic trust show
* (default)                                accept
172.30.1.1:5000                             accept
172.30.1.1:5000/production                 signed security@example.com
registry.redhat.io                         signed
security@redhat.com, security@redhat.com
```

4. You can further harden nodes by adding a global **reject** default trust:

```
$ atomic trust default reject

$ atomic trust show
* (default)                                reject
172.30.1.1:5000                             accept
172.30.1.1:5000/production                 signed security@example.com
registry.redhat.io                         signed
security@redhat.com, security@redhat.com
```

5. Optionally, review the **atomic** man page **man atomic-trust** for more configuration options.

3.8.5. Managing container logs

To prevent a container's log file, the `/var/lib/docker/containers/<hash>/<hash>-json.log` file on the node where the container is running, from increasing to a problematic size, you can configure Docker's **json-file** logging driver to restrict the size and number of log files.

Option	Purpose
<code>--log-opt max-size</code>	Sets the size at which a new log file is created.
<code>--log-opt max-file</code>	Sets the maximum number of log files to be kept per host.

1. To configure the log file, edit the `/etc/sysconfig/docker` file. For example, to set the maximum file size to 1 MB and always keep the last three log files, set the following options:

```
OPTIONS='--insecure-registry=172.30.0.0/16 --selinux-enabled --log-opt max-size=1M --log-opt max-file=3'
```

See Docker's documentation for additional information on how to [configure logging drivers](#).

2. Restart the Docker service:

```
# systemctl restart docker
```

3.8.6. Viewing available container logs

You can view the container logs in the `/var/lib/docker/containers/<hash>/` directory on the node where the container is running. For example:

```
# ls -lh
/var/lib/docker/containers/f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8/
total 2.6M
-rw-r--r--. 1 root root 5.6K Nov 24 00:12 config.json
-rw-r--r--. 1 root root 649K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log
-rw-r--r--. 1 root root 977K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-
json.log.1
-rw-r--r--. 1 root root 977K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-
json.log.2
-rw-r--r--. 1 root root 1.3K Nov 24 00:12 hostconfig.json
drwx-----. 2 root root    6 Nov 24 00:12 secrets
```

3.8.7. Blocking local volume usage

When a volume is provisioned using the **VOLUME** instruction in a *Dockerfile* or using the `docker run -v <volumename>` command, a host's storage space is used. Using this storage can lead to an unexpected out of space issue and can bring down the host.

In OpenShift Container Platform, users trying to run their own images risk filling the entire storage space on a node host. One solution to this issue is to prevent users from running images with volumes. This way, the only storage a user has access to can be limited, and the cluster administrator can assign storage quota.

Using **docker-novolume-plugin** solves this issue by disallowing starting a container with local volumes defined. In particular, the plug-in blocks **docker run** commands that contain:

- The **--volumes-from** option
- Images that have **VOLUME(s)** defined
- References to existing volumes that were provisioned with the **docker volume** command

The plug-in does not block references to bind mounts.

To enable **docker-novolume-plugin**, perform the following steps on each node host:

1. Install the **docker-novolume-plugin** package:

```
$ yum install docker-novolume-plugin
```

2. Enable and start the **docker-novolume-plugin** service:

```
$ systemctl enable docker-novolume-plugin
$ systemctl start docker-novolume-plugin
```

3. Edit the **/etc/sysconfig/docker** file and append the following to the **OPTIONS** list:

```
--authorization-plugin=docker-novolume-plugin
```

4. Restart the **docker** service:

```
$ systemctl restart docker
```

After you enable this plug-in, containers with local volumes defined fail to start and show the following error message:

```
runContainer: API error (500): authorization denied by plugin
docker-novolume-plugin: volumes are not allowed
```

3.9. RED HAT GLUSTER STORAGE SOFTWARE REQUIREMENTS

To access GlusterFS volumes, the **mount.glusterfs** command must be available on all schedulable nodes. For RPM-based systems, the **glusterfs-fuse** package must be installed:

```
# yum install glusterfs-fuse
```

This package comes installed on every RHEL system. However, it is recommended to update to the latest available version from Red Hat Gluster Storage. To do this, the following RPM repository must be enabled:

```
# subscription-manager repos --enable=rh-gluster-3-client-for-rhel-7-
server-rpms
```

If **glusterfs-fuse** is already installed on the nodes, ensure that the latest version is installed:

```
# yum update glusterfs-fuse
```

3.10. NEXT STEPS

After you finish preparing your hosts, if you are installing OpenShift Container Platform, [configure your inventory file](#). If you are installing a stand-alone registry, continue instead to [Installing a stand-alone registry](#).

CHAPTER 4. CONFIGURING YOUR INVENTORY FILE

4.1. CUSTOMIZING INVENTORY FILES FOR YOUR CLUSTER

Ansible inventory files describe the details about the hosts in your cluster and the cluster configuration details for your OpenShift Container Platform installation. The OpenShift Container Platform installation playbooks read your inventory file to know where and how to install OpenShift Container Platform across your set of hosts.



NOTE

See [Ansible documentation](#) for details about the format of an inventory file, including basic details about [YAML syntax](#).

When you install the **openshift-ansible** RPM package as described in [Host preparation](#), Ansible dependencies create a file at the default location of **/etc/ansible/hosts**. However, the file is simply the default Ansible example and has no variables related specifically to OpenShift Container Platform configuration. To successfully install OpenShift Container Platform, you *must* replace the default contents of the file with your own configuration based on your cluster topography and requirements.

The following sections describe commonly-used variables to set in your inventory file during cluster installation. Many of the Ansible variables described are optional. For development environments, you can accept the default values for the required parameters, but you must select appropriate values for them in production environments.

You can review [Example Inventory Files](#) for various examples to use as a starting point for your cluster installation.



NOTE

Images require a version number policy in order to maintain updates. See the [Image Version Tag Policy](#) section in the Architecture Guide for more information.

4.2. CONFIGURING CLUSTER VARIABLES

To assign global cluster environment variables during the Ansible installation, add them to the **[OSEv3:vars]** section of the **/etc/ansible/hosts** file. You must place each parameter value on a separate line. For example:

```
[OSEv3:vars]

openshift_master_identity_providers=[{'name': 'htpasswd_auth',
'login': 'true', 'challenge': 'true',
'kind': 'HTPasswdPasswordIdentityProvider',}]

openshift_master_default_subdomain=apps.test.example.com
```




IMPORTANT


If a parameter value in the Ansible inventory file contains special characters, such as `#`, `{` or `}`, you must double-escape the value (that is enclose the value in both single and double quotation marks). For example, to use `mypasswordwith###hashsigns` as a value for the variable `openshift_cloudprovider_openstack_password`, declare it as

`openshift_cloudprovider_openstack_password='"mypasswordwith###hashsigns"'` in the Ansible host inventory file.

The following tables describe global cluster variables for use with the Ansible installer:

Table 4.1. General Cluster Variables

Variable	Purpose
<code>ansible_ssh_user</code>	This variable sets the SSH user for the installer to use and defaults to root . This user must allow SSH-based authentication without requiring a password . If using SSH key-based authentication, then the key must be managed by an SSH agent.
<code>ansible_become</code>	If <code>ansible_ssh_user</code> is not root , this variable must be set to true and the user must be configured for passwordless sudo .
<code>debug_level</code>	<p>This variable sets which INFO messages are logged to the systemd-journald.service. Set one of the following:</p> <ul style="list-style-type: none"> • 0 to log errors and warnings only • 2 to log normal information (This is the default level.) • 4 to log debugging-level information • 6 to log API-level debugging information (request / response) • 8 to log body-level API debugging information <p>For more information on debug log levels, see Configuring Logging Levels.</p>

Variable	Purpose
openshift_clock_enabled	<p>Whether to enable Network Time Protocol (NTP) on cluster nodes. true by default.</p> <div>  <div> <p>IMPORTANT</p> <p>To prevent masters and nodes in the cluster from going out of sync, do not change the default value of this parameter.</p> </div> </div>
openshift_master_admission_plugin_config	<p>This variable sets the parameter and arbitrary JSON values as per the requirement in your inventory hosts file. For example:</p> <pre>openshift_master_admission_plugin_config= {"ClusterResourceOverride": {"configuration": {"apiVersion": "v1", "kind": "ClusterResourceOverrideConfig", "memoryRequestToLimitPercent": "25", "cpuRequestToLimitPercent": "25", "limitCPUToMemoryPercent": "200"}}}</pre>
openshift_master_audit_config	<p>This variable enables API service auditing. See Audit Configuration for more information.</p>
openshift_master_cluster_hostname	<p>This variable overrides the host name for the cluster, which defaults to the host name of the master.</p>
openshift_master_cluster_public_host_name	<p>This variable overrides the public host name for the cluster, which defaults to the host name of the master. If you use an external load balancer, specify the address of the external load balancer.</p> <p>For example:</p> <pre>openshift_master_cluster_public_hostname=openshift-ansible.public.example.com</pre>
openshift_master_cluster_method	<p>Optional. This variable defines the HA method when deploying multiple masters. Supports the native method. See Multiple Masters for more information.</p>

Variable	Purpose
<code>openshift_rolling_restart_mode</code>	This variable enables rolling restarts of HA masters (i.e., masters are taken down one at a time) when running the upgrade playbook directly . It defaults to services , which allows rolling restarts of services on the masters. It can instead be set to system , which enables rolling, full system restarts.
<code>openshift_master_identity_providers</code>	This variable sets the identity provider . The default value is Deny All . If you use a supported identity provider, configure OpenShift Container Platform to use it.
<code>openshift_master_named_certificates</code>	These variables are used to configure custom certificates which are deployed as part of the installation. See Configuring Custom Certificates for more information.
<code>openshift_master_overwrite_named_certificates</code>	
<code>openshift_hosted_router_certificate</code>	Provide the location of the custom certificates for the hosted router.
<code>openshift_master_ca_certificate</code>	Provide the single certificate and key that signs the OpenShift Container Platform certificates. See Redeploying a New or Custom OpenShift Container Platform CA
<code>openshift_additional_ca</code>	If the certificate for your <code>openshift_master_ca_certificate</code> parameter is signed by an intermediate certificate, provide the bundled certificate that contains the full chain of intermediate and root certificates for the CA. See Redeploying a New or Custom OpenShift Container Platform CA
<code>openshift_hosted_registry_cert_expire_days</code>	Validity of the auto-generated registry certificate in days. Defaults to 730 (2 years).
<code>openshift_ca_cert_expire_days</code>	Validity of the auto-generated CA certificate in days. Defaults to 1825 (5 years).
<code>openshift_node_cert_expire_days</code>	Validity of the auto-generated node certificate in days. Defaults to 730 (2 years).
<code>openshift_master_cert_expire_days</code>	Validity of the auto-generated master certificate in days. Defaults to 730 (2 years).
<code>etcd_ca_default_days</code>	Validity of the auto-generated external etcd certificates in days. Controls validity for etcd CA, peer, server and client certificates. Defaults to 1825 (5 years).

Variable	Purpose
openshift_certificate_expiry_warning_days	The amount of time the auto-generated certificates must be valid for an upgrade to proceed. Defaults to 365 (1 year).
openshift_certificate_expiry_fail_on_warn	Whether upgrade fails if the auto-generated certificates are not valid for the period specified by the openshift_certificate_expiry_warning_days parameter. Defaults to True .
os_firewall_use_firewalld	Set to true to use firewalld instead of the default iptables. Not available on RHEL Atomic Host. See the Configuring the Firewall section for more information.
openshift_master_session_name	These variables override defaults for session options in the OAuth configuration. See Configuring Session Options for more information.
openshift_master_session_max_seconds	
openshift_master_session_auth_secrets	
openshift_master_session_encryption_secrets	
openshift_master_image_policy_config	Sets imagePolicyConfig in the master configuration. See Image Configuration for details.
openshift_router_selector	Default node selector for automatically deploying router pods. See Configuring Node Host Labels for details.
openshift_registry_selector	Default node selector for automatically deploying registry pods. See Configuring Node Host Labels for details.
openshift_template_service_broker_namespaces	This variable enables the template service broker by specifying one or more namespaces whose templates will be served by the broker.
openshift_master_bootstrap_auto_approve	This variable enables TLS bootstrapping auto approval, which allows nodes to automatically join the cluster when provided a bootstrap credential. Set to true if you will enable the cluster auto-scaler on an Amazon Web Services (AWS) cluster. The default value is false .

Variable	Purpose
ansible_service_broker_node_selector	Default node selector for automatically deploying Ansible service broker pods, defaults {"node-role.kubernetes.io/infra": "true"} . See Configuring Node Host Labels for details.
template_service_broker_selector	Default node selector for automatically deploying template service broker pods, defaults {"node-role.kubernetes.io/infra": "true"} . See Configuring Node Host Labels for details.
osm_default_node_selector	This variable overrides the node selector that projects will use by default when placing pods, which is defined by the projectConfig.defaultNodeSelector field in the master configuration file. This defaults to node-role.kubernetes.io/compute=true if undefined.
openshift_docker_additional_registries	<p>OpenShift Container Platform adds the specified additional registry or registries to the docker configuration. These are the registries to search. If the registry requires access to a port other than 80, include the port number required in the form of <address>:<port>.</p> <p>For example:</p> <pre>openshift_docker_additional_registries=example.com:443</pre> <div>  <p>NOTE</p> <p>If you need to configure your cluster to use an alternate registry, set oreg_url rather than rely on openshift_docker_additional_registries.</p> </div>
openshift_docker_insecure_registries	OpenShift Container Platform adds the specified additional insecure registry or registries to the docker configuration. For any of these registries, secure sockets layer (SSL) is not verified. Can be set to the host name or IP address of the host. 0.0.0.0/0 is not a valid setting for the IP address.
openshift_docker_blocked_registries	OpenShift Container Platform adds the specified blocked registry or registries to the docker configuration. Block the listed registries. Setting this to all blocks everything not in the other variables.

Variable	Purpose
openshift_metrics_hawkular_hostname	This variable sets the host name for integration with the metrics console by overriding metricsPublicURL in the master configuration for cluster metrics. If you alter this variable, ensure the host name is accessible via your router.
openshift_clusterid	This variable is a cluster identifier unique to the AWS Availability Zone. Using this avoids potential issues in Amazon Web Services (AWS) with multiple zones or multiple clusters. See Labeling Clusters for AWS for details.
openshift_image_tag	Use this variable to specify a container image tag to install or configure.
openshift_pkg_version	Use this variable to specify an RPM version to install or configure.

**WARNING**

If you modify the **openshift_image_tag** or the **openshift_pkg_version** variables after the cluster is set up, then an upgrade can be triggered, resulting in downtime.

- If **openshift_image_tag** is set, its value is used for all hosts in system container environments, even those that have another version installed. If
- **openshift_pkg_version** is set, its value is used for all hosts in RPM-based environments, even those that have another version installed.

Table 4.2. Networking Variables

Variable	Purpose
openshift_master_default_subdomain	This variable overrides the default subdomain to use for exposed routes .
os_sdn_network_plugin_name	This variable configures which OpenShift SDN plug-in to use for the pod network, which defaults to redhat/openshift-ovs-subnet for the standard SDN plug-in. Set the variable to redhat/openshift-ovs-multitenant to use the multitenant SDN plug-in.

Variable	Purpose
<code>osm_cluster_network_cidr</code>	This variable overrides the SDN cluster network CIDR block. This is the network from which pod IPs are assigned. Specify a private block that does not conflict with existing network blocks in your infrastructure to which pods, nodes, or the master might require access. Defaults to 10.128.0.0/14 and cannot be arbitrarily re-configured after deployment, although certain changes to it can be made in the SDN master configuration .
<code>openshift_portal_net</code>	This variable configures the subnet in which services will be created in the OpenShift Container Platform SDN . Specify a private block that does not conflict with any existing network blocks in your infrastructure to which pods, nodes, or the master might require access to, or the installation will fail. Defaults to 172.30.0.0/16 , and cannot be re-configured after deployment. If changing from the default, avoid 172.17.0.0/16 , which the docker0 network bridge uses by default, or modify the docker0 network.
<code>osm_host_subnet_length</code>	This variable specifies the size of the per host subnet allocated for pod IPs by OpenShift Container Platform SDN . Defaults to 9 which means that a subnet of size /23 is allocated to each host; for example, given the default 10.128.0.0/14 cluster network, this will allocate 10.128.0.0/23, 10.128.2.0/23, 10.128.4.0/23, and so on. This cannot be re-configured after deployment.
<code>openshift_node_proxy_mode</code>	This variable specifies the service proxy mode to use: either iptables for the default, pure-iptables implementation, or userspace for the user space proxy.
<code>openshift_use_flannel</code>	This variable enables flannel as an alternative networking layer instead of the default SDN. If enabling flannel , disable the default SDN with the <code>openshift_use_openshift_sdn</code> variable. For more information, see Using Flannel .
<code>openshift_use_openshift_sdn</code>	Set to false to disable the OpenShift SDN plug-in.

4.3. CONFIGURING DEPLOYMENT TYPE

Various defaults used throughout the playbooks and roles used by the installer are based on the deployment type configuration (usually defined in an Ansible inventory file).

Ensure the `openshift_deployment_type` parameter in your inventory file's `[OSEv3:vars]` section is set to `openshift-enterprise` to install the OpenShift Container Platform variant:

```
[OSEv3:vars]
openshift_deployment_type=openshift-enterprise
```

4.4. CONFIGURING HOST VARIABLES

To assign environment variables to hosts during the Ansible installation, set them in the `/etc/ansible/hosts` file after the host entry in the `[masters]` or `[nodes]` sections. For example:

```
[masters]
ec2-52-6-179-239.compute-1.amazonaws.com openshift_public_hostname=ose3-
master.public.example.com
```

The following table describes variables for use with the Ansible installer that can be assigned to individual host entries:

Table 4.3. Host Variables

Variable	Purpose
<code>openshift_public_hostname</code>	This variable overrides the system's public host name. Use this for cloud installations, or for hosts on networks using a network address translation (NAT).
<code>openshift_public_ip</code>	This variable overrides the system's public IP address. Use this for cloud installations, or for hosts on networks using a network address translation (NAT).
<code>openshift_node_labels</code>	This variable is deprecated; see Defining Node Groups and Host Mappings for the current method of setting node labels.
<code>openshift_docker_options</code>	<p>This variable configures additional docker options in <code>/etc/sysconfig/docker</code>, such as options used in Managing Container Logs. It is recommended to use json-file.</p> <p>The following example shows the configuration of Docker to use the json-file log driver, where Docker will rotate between three 1 MB log files:</p> <pre>--log-driver json-file --log-opt max-size=1M --log-opt max-file=3"</pre>
<code>openshift_schedulable</code>	This variable configures whether the host is marked as a schedulable node, meaning that it is available for placement of new pods. See Configuring Schedulability on Masters .

Variable	Purpose
openshift_node_problem_detector_inst all	This variable is used to activate the Node Problem Detector . If set to false, the default , the Node Problem Detector is not installed or started.

4.5. DEFINING NODE GROUPS AND HOST MAPPINGS

Node configurations are [bootstrapped](#) from the master. When the node boots and services are started, the node checks if a **kubeconfig** and other node configuration files exist before joining the cluster. If they do not, the node pulls the configuration from the master, then joins the cluster.

This process replaces administrators having to manually maintain the node configuration uniquely on each node host. Instead, the contents of a node host's **/etc/origin/node/node-config.yaml** file are now provided by ConfigMaps sourced from the master.

4.5.1. Node ConfigMaps

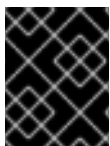
The Configmaps for defining the node configurations must be available in the **openshift-node** project. ConfigMaps are also now the authoritative definition for node labels; the old **openshift_node_labels** value is effectively ignored.

By default during a cluster installation, the installer creates the following default ConfigMaps:

- **node-config-master**
- **node-config-infra**
- **node-config-compute**

The following ConfigMaps are also created, which label nodes into multiple roles:

- **node-config-all-in-one**
- **node-config-master-infra**



IMPORTANT

You must not modify a node host's **/etc/origin/node/node-config.yaml** file. Any changes are overwritten by the configuration that is defined in the ConfigMap the node uses.

4.5.2. Node Group Definitions

After installing the latest **openshift-ansible** package, you can view the default set of node group definitions in YAML format in the **/usr/share/ansible/openshift-ansible/roles/openshift_facts/defaults/main.yml** file:

```
openshift_node_groups:
  - name: node-config-master ❶
    labels:
      - 'node-role.kubernetes.io/master=true' ❷
```

```

    edits: [] 3
  - name: node-config-infra
    labels:
      - 'node-role.kubernetes.io/infra=true'
    edits: []
  - name: node-config-compute
    labels:
      - 'node-role.kubernetes.io/compute=true'
    edits: []
  - name: node-config-master-infra
    labels:
      - 'node-role.kubernetes.io/infra=true,node-
role.kubernetes.io/master=true'
    edits: []
  - name: node-config-all-in-one
    labels:
      - 'node-role.kubernetes.io/infra=true,node-
role.kubernetes.io/master=true,node-role.kubernetes.io/compute=true'
    edits: []

```

- 1 Node group name.
- 2 List of node labels associated with the node group. See [Node Host Labels](#) for details.
- 3 Any edits to the node group's configuration.

If you do not set the **openshift_node_groups** variable in your inventory file's **[OSEv3:vars]** group, these default values are used. However, if you want to set custom node groups, you must define the entire **openshift_node_groups** structure, including all planned node groups, in your inventory file.

The **openshift_node_groups** value is not merged with the default values, and you must translate the YAML definitions into a Python dictionary. You can then use the **edits** field to modify any node configuration variables by specifying key-value pairs.



NOTE

See [Master and Node Configuration Files](#) for reference on configurable node variables.

For example, the following entry in an inventory file defines groups named **node-config-master**, **node-config-infra**, and **node-config-compute**.

```

openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels':
['node-role.kubernetes.io/infra=true']}, {'name': 'node-config-compute',
'labels': ['node-role.kubernetes.io/compute=true']}]

```

You can also define new node group names with other labels, the following entry in an inventory file defines groups named **node-config-master**, **node-config-infra**, **node-config-compute** and **node-config-compute-storage**.

```

openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels':
['node-role.kubernetes.io/infra=true']}, {'name': 'node-config-compute',

```

```
'labels': ['node-role.kubernetes.io/compute=true']], {'name': 'node-config-compute-storage', 'labels': ['node-role.kubernetes.io/compute-storage=true']}]}
```

When you set an entry in the inventory file, you can also edit the ConfigMap for a node group:

- You can use a list, such as modifying the **node-config-compute** to set **kubeletArguments.pods-per-core** to **20**:

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-role.kubernetes.io/compute=true'], 'edits': [{'key': 'kubeletArguments.pods-per-core', 'value': ['20']}]}]}
```

- You can use a list to modify multiple key value pairs, such as modifying the **node-config-compute** group to add two parameters to the **kubelet**:

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-role.kubernetes.io/compute=true'], 'edits': [{'key': 'kubeletArguments.experimental-allocatable-ignore-eviction', 'value': ['true']}, {'key': 'kubeletArguments.eviction-hard', 'value': ['memory.available<1Ki']}]}]}
```

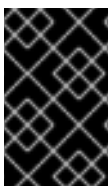
- You can also use a dictionary as value, such as modifying the **node-config-compute** group to set **perFSGroup** to **512Mi**:

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-role.kubernetes.io/compute=true'], 'edits': [{'key': 'volumeConfig.localQuota', 'value': {'perFSGroup': '512Mi'}}]}]
```

Whenever the **openshift_node_group.yml** playbook is run, the changes defined in the **edits** field will update the related ConfigMap (**node-config-compute** in this example), which will ultimately affect the node's configuration file on the host.

4.5.3. Mapping Hosts to Node Groups

To map which ConfigMap to use for which node host, all hosts defined in the **[nodes]** group of your inventory must be assigned to a *node group* using the **openshift_node_group_name** variable.



IMPORTANT

Setting **openshift_node_group_name** per host to a node group is required for all cluster installations whether you use the default node group definitions and ConfigMaps or are customizing your own.

The value of **openshift_node_group_name** is used to select the ConfigMap that configures each node. For example:

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
```

If other custom ConfigMaps have been defined in **openshift_node_groups** they can also be used. For example:

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
gluster[1:6].example.com openshift_node_group_name='node-config-compute-storage'
```

4.5.4. Node Host Labels

You can assign [Labels](#) to node hosts during cluster installation. You can use these labels to determine the placement of pods onto nodes using the [scheduler](#).

You must create your own custom node groups if you want to modify the default labels that are assigned to node hosts. You can no longer set the **openshift_node_labels** variable to change labels. See [Node Group Definitions](#) to modify the default node groups.

Other than **node-role.kubernetes.io/infra=true** (hosts using this group are also referred to as *dedicated infrastructure nodes* and discussed further in [Configuring Dedicated Infrastructure Nodes](#)), the actual label names and values are arbitrary and can be assigned however you see fit per your cluster's requirements.

4.5.4.1. Pod Schedulability on Masters

Configure all hosts that you designate as masters during the installation process as nodes. By doing so, the masters are configured as part of the [OpenShift SDN](#). You must add entries for the master hosts to the **[nodes]** section:

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
```

If you want to change the schedulability of a host post-installation, see [Marking Nodes as Unschedulable or Schedulable](#).

4.5.4.2. Pod Schedulability on Nodes

Masters are marked as schedulable nodes by default, so the default node selector is set by default during cluster installations. The default node selector is defined in the master configuration file's **projectConfig.defaultNodeSelector** field to determine which node projects will use by default when placing pods. It is set to **node-role.kubernetes.io/compute=true** unless overridden using the **osm_default_node_selector** variable.

**IMPORTANT**

If you accept the default node selector of **node-role.kubernetes.io/compute=true** during installation, ensure that you do not only have dedicated infrastructure nodes as the non-master nodes defined in your cluster. In that scenario, application pods fail to deploy because no nodes with the **node-role.kubernetes.io/compute=true** label are available to match the default node selector when scheduling pods for projects.

See [Setting the Cluster-wide Default Node Selector](#) for steps on adjusting this setting post-installation if needed.

4.5.4.3. Configuring Dedicated Infrastructure Nodes

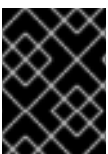
It is recommended for production environments that you maintain dedicated infrastructure nodes where the registry and router pods can run separately from pods used for user applications.

The **openshift_router_selector** and **openshift_registry_selector** Ansible settings determine the label selectors used when placing registry and router pods. They are set to **node-role.kubernetes.io/infra=true** by default:

```
# default selectors for router and registry services
# openshift_router_selector='node-role.kubernetes.io/infra=true'
# openshift_registry_selector='node-role.kubernetes.io/infra=true'
```

The registry and router are only able to run on node hosts with the **node-role.kubernetes.io/infra=true** label, which are then considered dedicated infrastructure nodes. Ensure that at least one node host in your OpenShift Container Platform environment has the **node-role.kubernetes.io/infra=true** label; you can use the default **node-config-infra**, which sets this label:

```
[nodes]
infra-node1.example.com openshift_node_group_name='node-config-infra'
```

**IMPORTANT**

If there is not a node in the **[nodes]** section that matches the selector settings, the default router and registry will be deployed as failed with **Pending** status.

If you do not intend to use OpenShift Container Platform to manage the registry and router, configure the following Ansible settings:

```
openshift_hosted_manage_registry=false
openshift_hosted_manage_router=false
```

If you use an image registry other than the default **registry.redhat.io**, you must [specify the registry](#) in the **/etc/ansible/hosts** file.

As described in [Configuring Schedulability on Masters](#), master hosts are marked schedulable by default. If you label a master host with **node-role.kubernetes.io/infra=true** and have no other dedicated infrastructure nodes, the master hosts must also be marked as schedulable. Otherwise, the registry and router pods cannot be placed anywhere.

You can use the default **node-config-master-infra** node group to achieve this:

```
[nodes]  
master.example.com openshift_node_group_name='node-config-master-infra'
```

4.6. CONFIGURING MASTER API PORT

To configure the default ports used by the master API, configure the following variables in the */etc/ansible/hosts* file:

Table 4.4. Master API Port

Variable	Purpose
openshift_master_api_port	This variable sets the port number to access the OpenShift Container Platform API.

For example:

```
openshift_master_api_port=3443
```

The web console port setting (**openshift_master_console_port**) must match the API server port (**openshift_master_api_port**).

4.7. CONFIGURING CLUSTER PRE-INSTALL CHECKS

Pre-install checks are a set of diagnostic tasks that run as part of the **openshift_health_checker** Ansible role. They run prior to an Ansible installation of OpenShift Container Platform, ensure that required inventory values are set, and identify potential issues on a host that can prevent or interfere with a successful installation.

The following table describes available pre-install checks that will run before every Ansible installation of OpenShift Container Platform:

Table 4.5. Pre-install Checks

Check Name	Purpose
memory_availability	This check ensures that a host has the recommended amount of memory for the specific deployment of OpenShift Container Platform. Default values have been derived from the latest installation documentation . A user-defined value for minimum memory requirements might be set by setting the openshift_check_min_host_memory_gb cluster variable in your inventory file.

Check Name	Purpose
disk_availability	<p>This check only runs on etcd, master, and node hosts. It ensures that the mount path for an OpenShift Container Platform installation has sufficient disk space remaining. Recommended disk values are taken from the latest installation documentation. A user-defined value for minimum disk space requirements might be set by setting openshift_check_min_host_disk_gb cluster variable in your inventory file.</p>
docker_storage	<p>Only runs on hosts that depend on the docker daemon (nodes and system container installations). Checks that docker's total usage does not exceed a user-defined limit. If no user-defined limit is set, docker's maximum usage threshold defaults to 90% of the total size available. The threshold limit for total percent usage can be set with a variable in your inventory file: max_thinpool_data_usage_percent=90. A user-defined limit for maximum thinpool usage might be set by setting the max_thinpool_data_usage_percent cluster variable in your inventory file.</p>
docker_storage_driver	<p>Ensures that the docker daemon is using a storage driver supported by OpenShift Container Platform. If the devicemapper storage driver is being used, the check additionally ensures that a loopback device is not being used. For more information, see Docker's Use the Device Mapper Storage Driver guide.</p>
docker_image_availability	<p>Attempts to ensure that images required by an OpenShift Container Platform installation are available either locally or in at least one of the configured container image registries on the host machine.</p>
package_version	<p>Runs on yum-based systems determining if multiple releases of a required OpenShift Container Platform package are available. Having multiple releases of a package available during an enterprise installation of OpenShift suggests that there are multiple yum repositories enabled for different releases, which might lead to installation problems. This check is skipped if the openshift_release variable is not defined in the inventory file.</p>

Check Name	Purpose
package_availability	Runs prior to RPM installations of OpenShift Container Platform. Ensures that RPM packages required for the current installation are available.
package_update	Checks whether a yum update or package installation will succeed, without actually performing it or running yum on the host.

To disable specific pre-install checks, include the variable **openshift_disable_check** with a comma-delimited list of check names in your inventory file. For example:

```
openshift_disable_check=memory_availability,disk_availability
```



NOTE

A similar set of health checks meant to run for diagnostics on existing clusters can be found in [Ansible-based Health Checks](#). Another set of checks for checking certificate expiration can be found in [Redeploying Certificates](#).

4.8. CONFIGURING A REGISTRY LOCATION

If you use an image registry other than the default at **registry.redhat.io**, specify the registry in the **/etc/ansible/hosts** file.

```
oreg_url=example.com/openshift3/ose-${component}:${version}
openshift_examples_modify_imagestreams=true
```

Table 4.6. Registry Variables

Variable	Purpose
oreg_url	Set to the alternate image location. Necessary if you are not using the default registry at registry.redhat.io . The default component inherits the image prefix and version from the oreg_url value. For the default registry, and registries that require authentication, you need to specify oreg_auth_user and oreg_auth_password .
openshift_examples_modify_imagestreams	Set to true if pointing to a registry other than the default. Modifies the image stream location to the value of oreg_url .

Variable	Purpose
oreg_auth_user	If oreg_url points to a registry requiring authentication, use the oreg_auth_user variable to provide your user name. You must also provide your password as the oreg_auth_password parameter value. If you use the default registry, specify a user that can access registry.redhat.io .
oreg_auth_password	If oreg_url points to a registry requiring authentication, use the oreg_auth_password variable to provide your password. You must also provide your user name as the oreg_auth_user parameter value. If you use the default registry, specify the password or token for that user.

**NOTE**

For more information on accessing the default registry, which requires authentication, see [Accessing and Configuring the Red Hat Registry](#)

For example:

```
oreg_url=example.com/openshift3/ose-${component}:${version}
openshift_examples_modify_imagestreams=true
```

4.9. CONFIGURING A REGISTRY ROUTE

To allow users to push and pull images to the internal container image registry from outside of the OpenShift Container Platform cluster, configure the registry route in the `/etc/ansible/hosts` file. By default, the registry route is **`docker-registry-default.router.default.svc.cluster.local`**.

Table 4.7. Registry Route Variables

Variable	Purpose
openshift_hosted_registry_routehost	Set to the value of the desired registry route. The route contains either a name that resolves to an infrastructure node where a router manages communication or the subdomain that you set as the default application subdomain wildcard value. For example, if you set the openshift_master_default_subdomain parameter to apps.example.com and .apps.example.com resolves to infrastructure nodes or a load balancer, you might use registry.apps.example.com as the registry route.

Variable	Purpose
openshift_hosted_registry_routecertificates	<p>Set the paths to the registry certificates. If you do not provide values for the certificate locations, certificates are generated. You can define locations for the following certificates:</p> <ul style="list-style-type: none"> • certfile • keyfile • cafile
openshift_hosted_registry_routetermination	<p>Set to one of the following values:</p> <ul style="list-style-type: none"> • Set to reencrypt to terminate encryption at the edge router and re-encrypt it with a new certificate supplied by the destination. • Set to passthrough to terminate encryption at the destination. The destination is responsible for decrypting traffic.

For example:

```
openshift_hosted_registry_routehost=<path>
openshift_hosted_registry_routetermination=reencrypt
openshift_hosted_registry_routecertificates= '{"certfile': '<path>/org-
cert.pem', 'keyfile': '<path>/org-privkey.pem', 'cafile': '<path>/org-
chain.pem'}"
```

4.10. CONFIGURING ROUTER SHARDING

[Router sharding](#) support is enabled by supplying the correct data to the inventory. The variable **openshift_hosted_routers** holds the data, which is in the form of a list. If no data is passed, then a default router is created. There are multiple combinations of router sharding. The following example supports routers on separate nodes:

```
openshift_hosted_routers=[{'name': 'router1', 'certificate': {'certfile':
'/path/to/certificate/abc.crt',
'keyfile': '/path/to/certificate/abc.key', 'cafile':
'/path/to/certificate/ca.crt'}, 'replicas': 1, 'serviceaccount': 'router',
'namespace': 'default', 'stats_port': 1936, 'edits': [], 'images':
'openshift3/ose-${component}:${version}', 'selector': 'type=router1',
'ports':
['80:80', '443:443']},
{'name': 'router2', 'certificate': {'certfile':
'/path/to/certificate/xyz.crt',
```

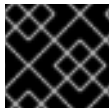
```
'keyfile': '/path/to/certificate/xyz.key', 'cafile':
'/path/to/certificate/ca.crt'}, 'replicas': 1, 'serviceaccount': 'router',
'namespace': 'default', 'stats_port': 1936, 'edits': [{'action': 'append',
'key': 'spec.template.spec.containers[0].env', 'value': {'name':
'ROUTE_LABELS',
'value': 'route=external'}}], 'images':
'openshift3/ose-${component}:${version}', 'selector': 'type=router2',
'ports':
['80:80', '443:443']}]}
```

4.11. CONFIGURING RED HAT GLUSTER STORAGE PERSISTENT STORAGE

Red Hat Gluster Storage can be configured to provide [persistent storage](#) and dynamic provisioning for OpenShift Container Platform. It can be used both containerized within OpenShift Container Platform (**converged mode**) and non-containerized on its own nodes (**independent mode**).

Additional information and examples, including the ones below, can be found at [Persistent Storage Using Red Hat Gluster Storage](#).

4.11.1. Configuring converged mode



IMPORTANT

See [converged mode Considerations](#) for specific host preparations and prerequisites.

1. In your inventory file, include the following variables in the **[OSEv3:vars]** section, and adjust them as required for your configuration:

```
[OSEv3:vars]
...
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=true
openshift_storage_glusterfs_block_host_vol_size=100
openshift_storage_glusterfs_block_storageclass=true
openshift_storage_glusterfs_block_storageclass_default=false
```

2. Add **glusterfs** in the **[OSEv3:children]** section to enable the **[glusterfs]** group:

```
[OSEv3:children]
masters
nodes
glusterfs
```

3. Add a **[glusterfs]** section with entries for each storage node that will host the GlusterFS storage. For each node, set **glusterfs_devices** to a list of raw block devices that will be completely managed as part of a GlusterFS cluster. There must be at least one device listed. Each device must be bare, with no partitions or LVM PVs. Specifying the variable takes the form:

```
<hostname_or_ip> glusterfs_devices='[ "</path/to/device1/>", "
```

```
</path/to/device2>", ... ]'
```

For example:

```
[glusterfs]
node11.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node12.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node13.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
```

4. Add the hosts listed under **[glusterfs]** to the **[nodes]** group:

```
[nodes]
...
node11.example.com openshift_node_group_name="node-config-compute"
node12.example.com openshift_node_group_name="node-config-compute"
node13.example.com openshift_node_group_name="node-config-compute"
```

4.11.2. Configuring independent mode

1. In your inventory file, include the following variables in the **[OSEv3:vars]** section, and adjust them as required for your configuration:

```
[OSEv3:vars]
...
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=true
openshift_storage_glusterfs_block_host_vol_size=100
openshift_storage_glusterfs_block_storageclass=true
openshift_storage_glusterfs_block_storageclass_default=false
openshift_storage_glusterfs_is_native=false
openshift_storage_glusterfs_heketi_is_native=true
openshift_storage_glusterfs_heketi_executor=ssh
openshift_storage_glusterfs_heketi_ssh_port=22
openshift_storage_glusterfs_heketi_ssh_user=root
openshift_storage_glusterfs_heketi_ssh_sudo=false
openshift_storage_glusterfs_heketi_ssh_keyfile="/root/.ssh/id_rsa"
```

2. Add **glusterfs** in the **[OSEv3:children]** section to enable the **[glusterfs]** group:

```
[OSEv3:children]
masters
nodes
glusterfs
```

3. Add a **[glusterfs]** section with entries for each storage node that will host the GlusterFS storage. For each node, set **glusterfs_devices** to a list of raw block devices that will be completely managed as part of a GlusterFS cluster. There must be at least one device listed. Each device must be bare, with no partitions or LVM PVs. Also, set **glusterfs_ip** to the IP address of the node. Specifying the variable takes the form:

```
<hostname_or_ip> glusterfs_ip=<ip_address> glusterfs_devices='[ "  
</path/to/device1/>', "</path/to/device2>", ... ]'
```

For example:

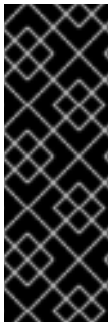
```
[glusterfs]  
gluster1.example.com glusterfs_ip=192.168.10.11 glusterfs_devices='[  
"/dev/xvdc", "/dev/xvdd" ]'  
gluster2.example.com glusterfs_ip=192.168.10.12 glusterfs_devices='[  
"/dev/xvdc", "/dev/xvdd" ]'  
gluster3.example.com glusterfs_ip=192.168.10.13 glusterfs_devices='[  
"/dev/xvdc", "/dev/xvdd" ]'
```

4.12. CONFIGURING AN OPENSIFT CONTAINER REGISTRY

An integrated [OpenShift Container Registry](#) can be deployed using the installer.

4.12.1. Configuring Registry Storage

If no registry storage options are used, the default OpenShift Container Registry is ephemeral and all data will be lost when the pod no longer exists.



IMPORTANT

Testing shows issues with using the NFS server on RHEL as storage backend for the container image registry. This includes the OpenShift Container Registry and Quay. Therefore, using NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift core components.

There are several options for enabling registry storage when using the advanced installer:

Option A: NFS Host Group

When the following variables are set, an NFS volume is created during cluster installation with the path **<nfs_directory>/<volume_name>** on the host in the **[nfs]** host group. For example, the volume path using these options is **/exports/registry**:

```
[OSEv3:vars]  
  
openshift_hosted_registry_storage_kind=nfs  
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']  
openshift_hosted_registry_storage_nfs_directory=/exports  
openshift_hosted_registry_storage_nfs_options='*(rw,root_squash)'  
openshift_hosted_registry_storage_volume_name=registry  
openshift_hosted_registry_storage_volume_size=10Gi
```

Option B: External NFS Host

To use an external NFS volume, one must already exist with a path of **<nfs_directory>/<volume_name>** on the storage host. The remote volume path using the following options is **nfs.example.com:/exports/registry**.

```
[OSEv3:vars]
```

```
openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_host=nfs.example.com
openshift_hosted_registry_storage_nfs_directory=/exports
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
```

Upgrading or Installing OpenShift Container Platform with NFS

Option C: OpenStack Platform

An OpenStack storage configuration must already exist.

```
[OSEv3:vars]
```

```
openshift_hosted_registry_storage_kind=openstack
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_openstack_filesystem=ext4
openshift_hosted_registry_storage_openstack_volumeID=3a650b4f-c8c5-4e0a-8ca5-eaee11f16c57
openshift_hosted_registry_storage_volume_size=10Gi
```

Option D: AWS or Another S3 Storage Solution

The simple storage solution (S3) bucket must already exist.

```
[OSEv3:vars]
```

```
#openshift_hosted_registry_storage_kind=object
#openshift_hosted_registry_storage_provider=s3
#openshift_hosted_registry_storage_s3_accesskey=access_key_id
#openshift_hosted_registry_storage_s3_secretkey=secret_access_key
#openshift_hosted_registry_storage_s3_bucket=bucket_name
#openshift_hosted_registry_storage_s3_region=bucket_region
#openshift_hosted_registry_storage_s3_chunksize=26214400
#openshift_hosted_registry_storage_s3_rootdirectory=/registry
#openshift_hosted_registry_pullthrough=true
#openshift_hosted_registry_acceptschema2=true
#openshift_hosted_registry_enforcequota=true
```

If you use a different S3 service, such as Minio or ExoScale, also add the region endpoint parameter:

```
openshift_hosted_registry_storage_s3_regionendpoint=https://myendpoint.example.com/
```

Option E: converged mode

Similar to [configuring converged mode](#), Red Hat Gluster Storage can be configured to provide storage for an OpenShift Container Registry during the initial installation of the cluster to offer redundant and reliable storage for the registry.



IMPORTANT

See [converged mode Considerations](#) for specific host preparations and prerequisites.

1. In your inventory file, set the following variable under **[OSEv3:vars]** section, and adjust them as required for your configuration:

```
[OSEv3:vars]
...
openshift_hosted_registry_storage_kind=glusterfs 1
openshift_hosted_registry_storage_volume_size=5Gi
openshift_hosted_registry_selector='node-
role.kubernetes.io/infra=true'
```

- 1 Running the integrated OpenShift Container Registry, on infrastructure nodes is recommended. Infrastructure node are nodes dedicated to running applications deployed by administrators to provide services for the OpenShift Container Platform cluster.

2. Add **glusterfs_registry** in the **[OSEv3:children]** section to enable the **[glusterfs_registry]** group:

```
[OSEv3:children]
masters
nodes
glusterfs_registry
```

3. Add a **[glusterfs_registry]** section with entries for each storage node that will host the GlusterFS storage. For each node, set **glusterfs_devices** to a list of raw block devices that will be completely managed as part of a GlusterFS cluster. There must be at least one device listed. Each device must be bare, with no partitions or LVM PVs. Specifying the variable takes the form:

```
<hostname_or_ip> glusterfs_devices='[ "</path/to/device1/>", "
</path/to/device2>", ... ]'
```

For example:

```
[glusterfs_registry]
node11.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node12.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node13.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
```

4. Add the hosts listed under **[glusterfs_registry]** to the **[nodes]** group:

```
[nodes]
...
node11.example.com openshift_node_group_name="node-config-infra"
node12.example.com openshift_node_group_name="node-config-infra"
node13.example.com openshift_node_group_name="node-config-infra"
```

Option F: Google Cloud Storage (GCS) bucket on Google Compute Engine (GCE)

A GCS bucket must already exist.

```
[OSEv3:vars]

openshift_hosted_registry_storage_provider=gcs
openshift_hosted_registry_storage_gcs_bucket=bucket01
```

```
openshift_hosted_registry_storage_gcs_keyfile=test.key
openshift_hosted_registry_storage_gcs_rootdirectory=/registry
```

Option G: vSphere Volume with vSphere Cloud Provider (VCP)

The vSphere Cloud Provider must be configured with a datastore accessible by the OpenShift Container Platform nodes.

When using vSphere volume for the registry, you must set the storage access mode to **ReadWriteOnce** and the replica count to **1**:

```
[OSEv3:vars]

openshift_hosted_registry_storage_kind=vsphere
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_annotations=[
'volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/vsphere-
volume']
openshift_hosted_registry_replicas=1
```

4.13. CONFIGURING GLOBAL PROXY OPTIONS

If your hosts require use of a HTTP or HTTPS proxy in order to connect to external hosts, there are many components that must be configured to use the proxy, including masters, Docker, and builds. Node services only connect to the master API requiring no external access and therefore do not need to be configured to use a proxy.

In order to simplify this configuration, the following Ansible variables can be specified at a cluster or host level to apply these settings uniformly across your environment.



NOTE

See [Configuring Global Build Defaults and Overrides](#) for more information on how the proxy environment is defined for builds.

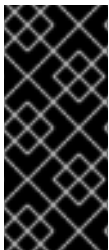
Table 4.8. Cluster Proxy Variables

Variable	Purpose
openshift_http_proxy	This variable specifies the HTTP_PROXY environment variable for masters and the Docker daemon.
openshift_https_proxy	This variable specifies the HTTPS_PROXY environment variable for masters and the Docker daemon.

Variable	Purpose
openshift_no_proxy	<p>This variable is used to set the NO_PROXY environment variable for masters and the Docker daemon. Provide a comma-separated list of host names, domain names, or wildcard host names that do not use the defined proxy. By default, this list is augmented with the list of all defined OpenShift Container Platform host names.</p> <p>The host names that do not use the defined proxy include:</p> <ul style="list-style-type: none"> • Master and node host names. You must include the domain suffix. • Other internal host names. You must include the domain suffix. • etcd IP addresses. You must provide the IP address because etcd access is managed by IP address. • The Docker registry IP address. • The Kubernetes IP address. This value is 172.30.0.1 by default and the openshift_portal_net parameter value if you provided one. • The cluster.local Kubernetes internal domain suffix. • The svc Kubernetes internal domain suffix.
openshift_generate_no_proxy_hosts	<p>This boolean variable specifies whether or not the names of all defined OpenShift hosts and *.cluster.local are automatically appended to the NO_PROXY list. Defaults to true; set it to false to override this option.</p>
openshift_builddefaults_http_proxy	<p>This variable defines the HTTP_PROXY environment variable inserted into builds using the BuildDefaults admission controller. If you do not define this parameter but define the openshift_http_proxy parameter, the openshift_http_proxy value is used. Set the openshift_builddefaults_http_proxy value to False to disable default http proxy for builds regardless of the openshift_http_proxy value.</p>

Variable	Purpose
<code>openshift_builddefaults_https_proxy</code>	This variable defines the HTTPS_PROXY environment variable inserted into builds using the BuildDefaults admission controller. If you do not define this parameter but define the <code>openshift_http_proxy</code> parameter, the <code>openshift_https_proxy</code> value is used. Set the <code>openshift_builddefaults_https_proxy</code> value to False to disable default https proxy for builds regardless of the <code>openshift_https_proxy</code> value.
<code>openshift_builddefaults_no_proxy</code>	This variable defines the NO_PROXY environment variable inserted into builds using the BuildDefaults admission controller. Set the <code>openshift_builddefaults_no_proxy</code> value to False to disable default no proxy settings for builds regardless of the <code>openshift_no_proxy</code> value.
<code>openshift_builddefaults_git_http_proxy</code>	This variable defines the HTTP proxy used by git clone operations during a build, defined using the BuildDefaults admission controller. Set the <code>openshift_builddefaults_git_http_proxy</code> value to False to disable default http proxy for git clone operations during a build regardless of the <code>openshift_http_proxy</code> value.
<code>openshift_builddefaults_git_https_proxy</code>	This variable defines the HTTPS proxy used by git clone operations during a build, defined using the BuildDefaults admission controller. Set the <code>openshift_builddefaults_git_https_proxy</code> value to False to disable default https proxy for git clone operations during a build regardless of the <code>openshift_https_proxy</code> value.

4.14. CONFIGURING THE FIREWALL



IMPORTANT

- If you are changing the default firewall, ensure that each host in your cluster is using the same firewall type to prevent inconsistencies.
- Do not use firewalld with the OpenShift Container Platform installed on Atomic Host. firewalld is not supported on Atomic host.



NOTE

While iptables is the default firewall, firewalld is recommended for new installations.

OpenShift Container Platform uses iptables as the default firewall, but you can configure your cluster to use firewalld during the install process.

Because iptables is the default firewall, OpenShift Container Platform is designed to have it configured automatically. However, iptables rules can break OpenShift Container Platform if not configured correctly. The advantages of firewalld include allowing multiple objects to safely share the firewall rules.

To use firewalld as the firewall for an OpenShift Container Platform installation, add the **os_firewall_use_firewalld** variable to the list of configuration variables in the Ansible host file at install:

```
[OSEv3:vars]
os_firewall_use_firewalld=True ❶
```

- ❶ Setting this variable to **true** opens the required ports and adds rules to the default zone, ensuring that firewalld is configured correctly.



NOTE

Using the firewalld default configuration comes with limited configuration options, and cannot be overridden. For example, while you can set up a storage network with interfaces in multiple zones, the interface that nodes communicate on must be in the default zone.

4.15. CONFIGURING SESSION OPTIONS

[Session options](#) in the OAuth configuration are configurable in the inventory file. By default, Ansible populates a **sessionSecretsFile** with generated authentication and encryption secrets so that sessions generated by one master can be decoded by the others. The default location is **/etc/origin/master/session-secrets.yaml**, and this file will only be re-created if deleted on all masters.

You can set the session name and maximum number of seconds with **openshift_master_session_name** and **openshift_master_session_max_seconds**:

```
openshift_master_session_name=ssn
openshift_master_session_max_seconds=3600
```

If provided, **openshift_master_session_auth_secrets** and **openshift_master_encryption_secrets** must be equal length.

For **openshift_master_session_auth_secrets**, used to authenticate sessions using HMAC, it is recommended to use secrets with 32 or 64 bytes:

```
openshift_master_session_auth_secrets=[ 'DONT+USE+THIS+SECRET+b4NV+pmZNS0' ]
```

For **openshift_master_encryption_secrets**, used to encrypt sessions, secrets must be 16, 24, or 32 characters long, to select AES-128, AES-192, or AES-256:

```
openshift_master_session_encryption_secrets=
[ 'DONT+USE+THIS+SECRET+b4NV+pmZNS0' ]
```

4.16. CONFIGURING CUSTOM CERTIFICATES

[Custom serving certificates](#) for the public host names of the OpenShift Container Platform API and [web console](#) can be deployed during cluster installation and are configurable in the inventory file.



NOTE

Configure custom certificates for the host name associated with the **publicMasterURL**, which you set as the **openshift_master_cluster_public_hostname** parameter value. Using a custom serving certificate for the host name associated with the **masterURL** (**openshift_master_cluster_hostname**) results in TLS errors because infrastructure components attempt to contact the master API using the internal **masterURL** host.

Certificate and key file paths can be configured using the **openshift_master_named_certificates** cluster variable:

```
openshift_master_named_certificates=[{"certfile": "/path/to/custom1.crt",
"keyfile": "/path/to/custom1.key", "cafile": "/path/to/custom-ca1.crt"}]
```

File paths must be local to the system where Ansible will be run. Certificates are copied to master hosts and are deployed in the **/etc/origin/master/named_certificates/** directory.

Ansible detects a certificate's **Common Name** and **Subject Alternative Names**. Detected names can be overridden by providing the **"names"** key when setting **openshift_master_named_certificates**:

```
openshift_master_named_certificates=[{"certfile": "/path/to/custom1.crt",
"keyfile": "/path/to/custom1.key", "names": ["public-master-host.com"],
"cafile": "/path/to/custom-ca1.crt"}]
```

Certificates configured using **openshift_master_named_certificates** are cached on masters, meaning that each additional Ansible run with a different set of certificates results in all previously deployed certificates remaining in place on master hosts and in the master configuration file.

If you want to overwrite **openshift_master_named_certificates** with the provided value (or no value), specify the **openshift_master_overwrite_named_certificates** cluster variable:

```
openshift_master_overwrite_named_certificates=true
```

For a more complete example, consider the following cluster variables in an inventory file:

```
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lb-internal.openshift.com
openshift_master_cluster_public_hostname=custom.openshift.com
```

To overwrite the certificates on a subsequent Ansible run, set the following parameter values:

```
openshift_master_named_certificates=[{"certfile":
"/root/STAR.openshift.com.crt", "keyfile": "/root/STAR.openshift.com.key",
"names": ["custom.openshift.com"]}]
openshift_master_overwrite_named_certificates=true
```

4.17. CONFIGURING CERTIFICATE VALIDITY

By default, the certificates used to govern the etcd, master, and kubelet expire after two to five years. The validity (length in days until they expire) for the auto-generated registry, CA, node, and master certificates can be configured during installation using the following variables (default values shown):

```
[OSEv3:vars]

openshift_hosted_registry_cert_expire_days=730
openshift_ca_cert_expire_days=1825
openshift_node_cert_expire_days=730
openshift_master_cert_expire_days=730
etcd_ca_default_days=1825
```

These values are also used when [redeploying certificates](#) via Ansible post-installation.

4.18. CONFIGURING CLUSTER MONITORING

Prometheus Cluster Monitoring is set to automatically deploy. To prevent its automatic deployment, set the following:

```
[OSEv3:vars]

openshift_cluster_monitoring_operator_install=false
```

For more information on Prometheus Cluster Monitoring and its configuration, see [Prometheus Cluster Monitoring documentation](#).

4.19. CONFIGURING CLUSTER METRICS

Cluster metrics are not set to automatically deploy. Set the following to enable cluster metrics during cluster installation:

```
[OSEv3:vars]

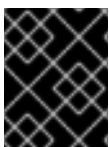
openshift_metrics_install_metrics=true
```

The metrics public URL can be set during cluster installation using the `openshift_metrics_hawkular_hostname` Ansible variable, which defaults to:

```
https://hawkular-metrics.
{{openshift_master_default_subdomain}}/hawkular/metrics
```

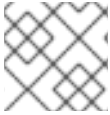
If you alter this variable, ensure the host name is accessible via your router.

```
openshift_metrics_hawkular_hostname=hawkular-metrics.
{{openshift_master_default_subdomain}}
```



IMPORTANT

In accordance with upstream Kubernetes rules, metrics can be collected only on the default interface of `eth0`.

**NOTE**

You must set an **openshift_master_default_subdomain** value to deploy metrics.

4.19.1. Configuring Metrics Storage

The **openshift_metrics_cassandra_storage_type** variable must be set in order to use persistent storage for metrics. If **openshift_metrics_cassandra_storage_type** is not set, then cluster metrics data is stored in an **emptyDir** volume, which will be deleted when the Cassandra pod terminates.

**IMPORTANT**

Testing shows issues with using the NFS server on RHEL as storage backend for the container image registry. This includes Cassandra for metrics storage. Therefore, using NFS to back PVs used by core services is not recommended.

Cassandra is designed to provide redundancy via multiple independent, instances. For this reason, using NFS or a SAN for data directories is an antipattern and is not recommended.

However, NFS/SAN implementations on the marketplace might not have issues backing or providing storage to this component. Contact the individual NFS/SAN implementation vendor for more information on any testing that was possibly completed against these OpenShift core components.

There are three options for enabling cluster metrics storage during cluster installation:

Option A: Dynamic

If your OpenShift Container Platform environment supports [dynamic volume provisioning](#) for your cloud provider, use the following variable:

```
[OSEv3:vars]

openshift_metrics_cassandra_storage_type=dynamic
```

If there are multiple default dynamically provisioned volume types, such as gluster-storage and glusterfs-storage-block, you can specify the provisioned volume type by variable. For example, **openshift_metrics_cassandra_pvc_storage_class_name=glusterfs-storage-block**.

Check [Volume Configuration](#) for more information on using **DynamicProvisioningEnabled** to enable or disable dynamic provisioning.

Option B: NFS Host Group

When the following variables are set, an NFS volume is created during cluster installation with path **<nfs_directory>/<volume_name>** on the host in the **[nfs]** host group. For example, the volume path using these options is **/exports/metrics**:

```
[OSEv3:vars]

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=[ 'ReadWriteOnce' ]
openshift_metrics_storage_nfs_directory=/exports
```

```
openshift_metrics_storage_nfs_options='*(rw,root_squash)'
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=10Gi
```

Option C: External NFS Host

To use an external NFS volume, one must already exist with a path of **<nfs_directory>/<volume_name>** on the storage host.

```
[OSEv3:vars]

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_host=nfs.example.com
openshift_metrics_storage_nfs_directory=/exports
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=10Gi
```

The remote volume path using the following options is **nfs.example.com:/exports/metrics**.

Upgrading or Installing OpenShift Container Platform with NFS

The use of NFS for the core OpenShift Container Platform components is not recommended, as NFS (and the NFS Protocol) does not provide the proper consistency needed for the applications that make up the OpenShift Container Platform infrastructure.

As a result, the installer and update playbooks require an option to enable the use of NFS with core infrastructure components.

```
# Enable unsupported configurations, things that will yield a partially
# functioning cluster but would not be supported for production use
#openshift_enable_unsupported_configurations=false
```

If you see the following messages when upgrading or installing your cluster, then an additional step is required.

```
TASK [Run variable sanity checks]
*****
fatal: [host.example.com]: FAILED! => {"failed": true, "msg":
"last_checked_host: host.example.com, last_checked_var:
openshift_hosted_registry_storage_kind;nfs is an unsupported type for
openshift_hosted_registry_storage_kind.
openshift_enable_unsupported_configurations=True mustbe specified to
continue with this configuration."}
```

In your Ansible inventory file, specify the following parameter:

```
[OSEv3:vars]
openshift_enable_unsupported_configurations=True
```

4.20. CONFIGURING CLUSTER LOGGING

Cluster logging is not set to automatically deploy by default. Set the following to enable cluster logging during cluster installation:

```
[OSEv3:vars]
```

```
openshift_logging_install_logging=true
```

4.20.1. Configuring Logging Storage

The `openshift_logging_es_pvc_dynamic` variable must be set in order to use persistent storage for logging. If `openshift_logging_es_pvc_dynamic` is not set, then cluster logging data is stored in an `emptyDir` volume, which will be deleted when the Elasticsearch pod terminates.

IMPORTANT

Testing shows issues with using the NFS server on RHEL as storage backend for the container image registry. This includes Elasticsearch for logging storage. Therefore, using NFS to back PVs used by core services is not recommended.

Due to Elasticsearch not implementing a custom deletionPolicy, the use of NFS storage as a volume or a persistent volume is not supported for Elasticsearch storage, as Lucene and the default deletionPolicy, relies on file system behavior that NFS does not supply. Data corruption and other problems can occur.

NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing they might have performed against these OpenShift core components.

There are three options for enabling cluster logging storage during cluster installation:

Option A: Dynamic

If your OpenShift Container Platform environment supports [dynamic volume provisioning](#) for your cloud provider, use the following variable:

```
[OSEv3:vars]

openshift_logging_es_pvc_dynamic=true
```

If there are multiple default dynamically provisioned volume types, such as `gluster-storage` and `glusterfs-storage-block`, you can specify the provisioned volume type by variable. For example, **`openshift_logging_es_pvc_storage_class_name=glusterfs-storage-block`**.

Check [Volume Configuration](#) for more information on using `DynamicProvisioningEnabled` to enable or disable dynamic provisioning.

Option B: NFS Host Group

When the following variables are set, an NFS volume is created during cluster installation with path `<nfs_directory>/<volume_name>` on the host in the `[nfs]` host group. For example, the volume path using these options is `/exports/logging`:

```
[OSEv3:vars]

openshift_logging_storage_kind=nfs
openshift_logging_storage_access_modes=['ReadWriteOnce']
openshift_logging_storage_nfs_directory=/exports
openshift_logging_storage_nfs_options='*(rw,root_squash)'
openshift_logging_storage_volume_name=logging
openshift_logging_storage_volume_size=10Gi
```


Option C: External NFS Host

To use an external NFS volume, one must already exist with a path of `<nfs_directory>/<volume_name>` on the storage host.

```
[OSEv3:vars]

openshift_logging_storage_kind=nfs
openshift_logging_storage_access_modes=['ReadWriteOnce']
openshift_logging_storage_host=nfs.example.com
openshift_logging_storage_nfs_directory=/exports
openshift_logging_storage_volume_name=logging
openshift_logging_storage_volume_size=10Gi
```

The remote volume path using the following options is ***nfs.example.com:/exports/logging***.

Upgrading or Installing OpenShift Container Platform with NFS

The use of NFS for the core OpenShift Container Platform components is not recommended, as NFS (and the NFS Protocol) does not provide the proper consistency needed for the applications that make up the OpenShift Container Platform infrastructure.

As a result, the installer and update playbooks require an option to enable the use of NFS with core infrastructure components.

```
# Enable unsupported configurations, things that will yield a partially
# functioning cluster but would not be supported for production use
#openshift_enable_unsupported_configurations=false
```

If you see the following messages when upgrading or installing your cluster, then an additional step is required.

```
TASK [Run variable sanity checks]
*****
fatal: [host.example.com]: FAILED! => {"failed": true, "msg":
"last_checked_host: host.example.com, last_checked_var:
openshift_hosted_registry_storage_kind;nfs is an unsupported type for
openshift_hosted_registry_storage_kind.
openshift_enable_unsupported_configurations=True mustbe specified to
continue with this configuration."}
```

In your Ansible inventory file, specify the following parameter:

```
[OSEv3:vars]
openshift_enable_unsupported_configurations=True
```

4.21. CUSTOMIZING SERVICE CATALOG OPTIONS

The [service catalog](#) is enabled by default during installation. Enabling the service broker allows you to register service brokers with the catalog. When the service catalog is enabled, the OpenShift Ansible broker and template service broker are both installed as well; see [Configuring the OpenShift Ansible Broker](#) and [Configuring the Template Service Broker](#) for more information. If you disable the service catalog, the OpenShift Ansible broker and template service broker are not installed.

To disable automatic deployment of the service catalog, set the following cluster variable in your inventory file:

```
openshift_enable_service_catalog=false
```

If you use your own registry, you must add:

- **openshift_service_catalog_image_prefix:** When pulling the service catalog image, force the use of a specific prefix (for example, **registry**). You must provide the full registry name up to the image name.
- **openshift_service_catalog_image_version:** When pulling the service catalog image, force the use of a specific image version.

For example:

```
openshift_service_catalog_image="docker-
registry.default.example.com/openshift/ose-service-catalog:${version}"
openshift_service_catalog_image_prefix="docker-registry-
default.example.com/openshift/ose-"
openshift_service_catalog_image_version="v3.9.30"
template_service_broker_selector={"role":"infra"}
```

4.21.1. Configuring the OpenShift Ansible Broker

The [OpenShift Ansible broker](#) (OAB) is enabled by default during installation.

If you do not want to install the OAB, set the **ansible_service_broker_install** parameter value to **false** in the inventory file:

```
ansible_service_broker_install=false
```

Table 4.9. Service broker customization variables

Variable	Purpose
openshift_service_catalog_image_prefix	Specify the prefix for the service catalog component image.

4.21.1.1. Configuring Persistent Storage for the OpenShift Ansible Broker

The OAB deploys its own etcd instance separate from the etcd used by the rest of the OpenShift Container Platform cluster. The OAB's etcd instance requires separate storage using persistent volumes (PVs) to function. If no PV is available, etcd will wait until the PV can be satisfied. The OAB application will enter a **CrashLoop** state until its etcd instance is available.

Some Ansible playbook bundles (APBs) also require a PV for their own usage in order to deploy. For example, each of the database APBs have two plans: the Development plan uses ephemeral storage and does not require a PV, while the Production plan is persisted and does require a PV.

APB	PV Required?
postgresql-apb	Yes, but only for the Production plan

APB	PV Required?
mysql-apb	Yes, but only for the Production plan
mariadb-apb	Yes, but only for the Production plan
mediawiki-apb	Yes

To configure persistent storage for the OAB:



NOTE

The following example shows usage of an NFS host to provide the required PVs, but [other persistent storage providers](#) can be used instead.

1. In your inventory file, add **nfs** to the **[OSEv3:children]** section to enable the **[nfs]** group:

```
[OSEv3:children]
masters
nodes
nfs
```

2. Add a **[nfs]** group section and add the host name for the system that will be the NFS host:

```
[nfs]
master1.example.com
```

3. Add the following in the **[OSEv3:vars]** section:

```
openshift_hosted_etcd_storage_kind=nfs
openshift_hosted_etcd_storage_nfs_options="*
(rw,root_squash, sync,no_wdelay)"
openshift_hosted_etcd_storage_nfs_directory=/opt/osev3-etcd 1
openshift_hosted_etcd_storage_volume_name=etcd-vol2 2
openshift_hosted_etcd_storage_access_modes=["ReadWriteOnce"]
openshift_hosted_etcd_storage_volume_size=1G
openshift_hosted_etcd_storage_labels={'storage': 'etcd'}
```

- 1 2** An NFS volume will be created with path **<nfs_directory>/<volume_name>** on the host in the **[nfs]** group. For example, the volume path using these options is ***/opt/osev3-etcd/etcd-vol2***.

These settings create a persistent volume that is attached to the OAB's etcd instance during cluster installation.

4.21.1.2. Configuring the OpenShift Ansible Broker for Local APB Development

In order to do [APB development](#) with the OpenShift Container Registry in conjunction with the OAB, a whitelist of images the OAB can access must be defined. If a whitelist is not defined, the broker will ignore APBs and users will not see any APBs available.

By default, the whitelist is empty so that a user cannot add APB images to the broker without a cluster administrator configuring the broker. To whitelist all images that end in **-apb**:

1. In your inventory file, add the following to the **[OSEv3:vars]** section:

```
ansible_service_broker_local_registry_whitelist=['.*-apb$']
```

4.21.2. Configuring the Template Service Broker

The [template service broker](#) (TSB) is enabled by default during installation.

If you do not want to install the TSB, set the **template_service_broker_install** parameter value to **false**:

```
template_service_broker_install=false
```

To configure the TSB, one or more projects must be defined as the broker's source namespace(s) for loading templates and image streams into the service catalog. Set the source projects by modifying the following in your inventory file's **[OSEv3:vars]** section:

```
openshift_template_service_broker_namespaces=['openshift','myproject']
```

By default, the TSB uses the nodeselector **{"node-role.kubernetes.io/infra":"true"}** for deploying its pods. You can set a different nodeselector in your inventory file's **[OSEv3:vars]** section:

```
template_service_broker_selector={"node-role.kubernetes.io/infra":"true"}
```

Table 4.10. Template service broker customization variables

Variable	Purpose
template_service_broker_prefix	Specify the prefix for the template service broker component image.
ansible_service_broker_image_prefix	Specify the prefix for the ansible service broker component image.

4.22. CONFIGURING WEB CONSOLE CUSTOMIZATION

The following Ansible variables set master configuration options for customizing the web console. See [Customizing the Web Console](#) for more details on these customization options.

Table 4.11. Web Console Customization Variables

Variable	Purpose
openshift_web_console_install	Determines whether to install the web console. Can be set to true or false . Defaults to true .
openshift_web_console_prefix	Specify the prefix for the web console images.

Variable	Purpose
<code>openshift_master_logout_url</code>	Sets <code>clusterInfo.logoutPublicURL</code> in the web console configuration. See Changing the Logout URL for details. Example value: <code>https://example.com/logout</code>
<code>openshift_web_console_extension_script_urls</code>	Sets <code>extensions.scriptURLs</code> in the web console configuration. See Loading Extension Scripts and Stylesheets for details. Example value: <code>['https://example.com/scripts/menu-customization.js', 'https://example.com/scripts/nav-customization.js']</code>
<code>openshift_web_console_extension_stylesheet_urls</code>	Sets <code>extensions.stylesheetURLs</code> in the web console configuration. See Loading Extension Scripts and Stylesheets for details. Example value: <code>['https://example.com/styles/logo.css', 'https://example.com/styles/custom-styles.css']</code>
<code>openshift_master_oauth_template</code>	Sets the OAuth template in the master configuration. See Customizing the Login Page for details. Example value: <code>['/path/to/login-template.html']</code>
<code>openshift_master_metrics_public_url</code>	Sets <code>metricsPublicURL</code> in the master configuration. See Setting the Metrics Public URL for details. Example value: <code>https://hawkular-metrics.example.com/hawkular/metrics</code>
<code>openshift_master_logging_public_url</code>	Sets <code>loggingPublicURL</code> in the master configuration. See Kibana for details. Example value: <code>https://kibana.example.com</code>
<code>openshift_web_console_inactivity_timeout_minutes</code>	Configure the web console to log the user out automatically after a period of inactivity. Must be a whole number greater than or equal to 5, or 0 to disable the feature. Defaults to 0 (disabled).
<code>openshift_web_console_cluster_resource_overrides_enabled</code>	Boolean value indicating if the cluster is configured for overcommit. When true , the web console will hide fields for CPU request, CPU limit, and memory request when editing resource limits because you must set these values with the cluster resource override configuration.
<code>openshift_web_console_enable_context_selector</code>	Enable the context selector in the web console and admin console mastheads for quickly switching between the two consoles. Defaults to true when both consoles are installed.

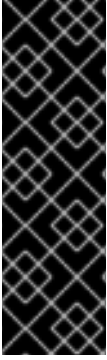
4.23. CONFIGURING THE CLUSTER CONSOLE

The cluster console is an additional web interface like the web console, but focused on admin tasks. The cluster console supports many of the same common OpenShift Container Platform resources as the web console, but it also allows you to view metrics about the cluster and manage cluster-scoped resources such as nodes, persistent volumes, cluster roles, and custom resource definitions. The following variables can be used to customize the cluster console.

Table 4.12. Cluster Console Customization Variables

Variable	Purpose
<code>openshift_console_install</code>	Determines whether to install the cluster console. Can be set to true or false . Defaults to true .
<code>openshift_console_hostname</code>	Sets the host name of the cluster console. Defaults to console . <code><openshift_master_default_subdomain></code> . If you alter this variable, ensure the host name is accessible via your router.
<code>openshift_console_cert</code>	Optional certificate to use for the cluster console route. This is only needed if using a custom host name.
<code>openshift_console_key</code>	Optional key to use for the cluster console route. This is only needed if using a custom host name.
<code>openshift_console_ca</code>	Optional CA to use for the cluster console route. This is only needed if using a custom host name.
<code>openshift_base_path</code>	Optional base path for the cluster console. If set, it should begin and end with a slash like <code>/console/</code> . Defaults to <code>/</code> (no base path).
<code>openshift_console_auth_ca_file</code>	Optional CA file to use to connect to the OAuth server. Defaults to <code>/var/run/secrets/kubernetes.io/serviceaccount/ca.crt</code> . Typically this does not need to be changed.

4.24. CONFIGURING THE OPERATOR LIFECYCLE MANAGER



IMPORTANT

The Operator Framework is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for 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 on Red Hat Technology Preview features support scope, see <https://access.redhat.com/support/offerings/techpreview/>.

The Technology Preview [Operator Framework](#) includes the Operator Lifecycle Manager (OLM). You can optionally install the OLM during cluster installation by setting the following variables in your inventory file:



NOTE

Alternatively, the Technology Preview Operator Framework can be installed after cluster installation. See [Installing Operator Lifecycle Manager using Ansible](#) for separate instructions.

1. Add the **openshift_enable_olm** variable in the **[OSEv3:vars]** section, setting it to **true**:

```
openshift_enable_olm=true
```

2. Add the **openshift_additional_registry_credentials** variable in the **[OSEv3:vars]** section, setting credentials required to pull the Operator containers:

```
openshift_additional_registry_credentials=[
  {'host':'registry.connect.redhat.com','user':'<your_user_name>','password':'<your_password>','test_image':'mongodb/enterprise-operator:0.3.2'}]
```

Set **user** and **password** to the credentials that you use to log in to the Red Hat Customer Portal at <https://access.redhat.com>.

The **test_image** represents an image that will be used to test the credentials you provided.

After your cluster installation has completed successful, see [Launching your first Operator](#) for further steps on using the OLM as a cluster administrator during this Technology Preview phase.

CHAPTER 5. EXAMPLE INVENTORY FILES

5.1. OVERVIEW

After getting to know the basics of [configuring your own inventory file](#), you can review the following example inventories which describe various environment topographies, including [using multiple masters](#) for high availability. You can choose an example that matches your requirements, modify it to match your own environment, and use it as your inventory file when [running the installation](#).



IMPORTANT

The following example inventories use the default set of node groups when setting **openshift_node_group_name** per host in the **[nodes]** group. To define and use your own custom node group definitions, set the **openshift_node_groups** variable in the inventory file. See [Defining Node Groups and Host Mappings](#) for details.

5.2. SINGLE MASTER EXAMPLES

You can configure an environment with a single master and multiple nodes, and either a single or multiple number of etcd hosts.



NOTE

Moving from a single master cluster to multiple masters after installation is not supported.

5.2.1. Single Master, Single etcd, and Multiple Nodes

The following table describes an example environment for a single [master](#) (with a single etcd instance running as a static pod on the same host), two [nodes](#) for hosting user applications, and two nodes with the **node-role.kubernetes.io/infra=true** label for hosting [dedicated infrastructure](#):

Host Name	Component/Role(s) to Install
master.example.com	Master, etcd, and node
node1.example.com	Compute node
node2.example.com	
infra-node1.example.com	Infrastructure node
infra-node2.example.com	

You can see these example hosts present in the **[masters]**, **[etcd]**, and **[nodes]** sections of the following example inventory file:

Single Master, Single etcd, and Multiple Nodes Inventory File

```
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
```



```

masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring a
password
ansible_ssh_user=root

# If ansible_ssh_user is not root, ansible_become must be set to true
#ansible_become=true

openshift_deployment_type=openshift-enterprise
oreg_url=example.com/openshift3/ose-${component}:${version}
oreg_auth_user=${user_name}
oreg_auth_password=${password}
openshift_examples_modify_imagestreams=true

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login':
'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
master.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'

```



IMPORTANT

See [Configuring Node Host Labels](#) to ensure you understand the default node selector requirements and node label considerations beginning in OpenShift Container Platform 3.9.

To use this example, modify the file to match your environment and specifications, and save it as ***/etc/ansible/hosts***.

5.2.2. Single Master, Multiple etcd, and Multiple Nodes

The following table describes an example environment for a single [master](#), three [etcd](#) hosts, two [nodes](#) for hosting user applications, and two nodes with the **node-role.kubernetes.io/infra=true** label for hosting [dedicated infrastructure](#):

Host Name	Component/Role(s) to Install
master.example.com	Master and node
etcd1.example.com	etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	Compute node
node2.example.com	
infra-node1.example.com	Dedicated infrastructure node
infra-node2.example.com	

You can see these example hosts present in the **[masters]**, **[nodes]**, and **[etcd]** sections of the following example inventory file:

Single Master, Multiple etcd, and Multiple Nodes Inventory File

```
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
masters
nodes
etcd

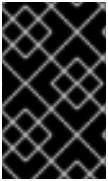
# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
oreg_url=example.com/openshift3/ose-${component}:${version}
oreg_auth_user=${user_name}
oreg_auth_password=${password}
openshift_examples_modify_imagestreams=true

# uncomment the following to enable htpasswd authentication; defaults to
# DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login':
# 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com
```

```
# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```



IMPORTANT

See [Configuring Node Host Labels](#) to ensure you understand the default node selector requirements and node label considerations beginning in OpenShift Container Platform 3.9.

To use this example, modify the file to match your environment and specifications, and save it as */etc/ansible/hosts*.

5.3. MULTIPLE MASTERS EXAMPLES

You can configure an environment with multiple masters, multiple etcd hosts, and multiple nodes. Configuring [multiple masters for high availability](#) (HA) ensures that the cluster has no single point of failure.



NOTE

Moving from a single master cluster to multiple masters after installation is not supported.

When configuring multiple masters, the cluster installation process supports the **native** high availability (HA) method. This method leverages the native HA master capabilities built into OpenShift Container Platform and can be combined with any load balancing solution.

If a host is defined in the **[lb]** section of the inventory file, Ansible installs and configures HAProxy automatically as the load balancing solution. If no host is defined, it is assumed you have pre-configured an external load balancing solution of your choice to balance the master API (port 8443) on all master hosts.



NOTE

This HAProxy load balancer is intended to demonstrate the API server's HA mode and is not recommended for production environments. If you are deploying to a cloud provider, Red Hat recommends deploying a cloud-native TCP-based load balancer or take other steps to provide a highly available load balancer.

For an external load balancing solution, you must have:

- A pre-created load balancer virtual IP (VIP) configured for SSL passthrough.
- A VIP listening on the port specified by the [openshift_master_api_port](#) value (8443 by default) and proxying back to all master hosts on that port.
- A domain name for VIP registered in DNS.

- o The domain name will become the value of both **openshift_master_cluster_public_hostname** and **openshift_master_cluster_hostname** in the OpenShift Container Platform installer.

See [the External Load Balancer Integrations example in Github](#) for more information. For more on the high availability master architecture, see [Kubernetes Infrastructure](#).



NOTE

The cluster installation process does not currently support multiple HAProxy load balancers in an active-passive setup. See the [Load Balancer Administration documentation](#) for post-installation amendments.

To configure multiple masters, refer to [Multiple Masters with Multiple etcd](#)

5.3.1. Multiple Masters Using Native HA with External Clustered etcd

The following describes an example environment for three [masters](#) using the **native** HA method:, one HAProxy load balancer, three [etcd](#) hosts, two [nodes](#) for hosting user applications, and two nodes with the **node-role.kubernetes.io/infra=true** label for hosting [dedicated infrastructure](#):

Host Name	Component/Role(s) to Install
master1.example.com	Master (clustered using native HA) and node
master2.example.com	
master3.example.com	
lb.example.com	HAProxy to load balance API master endpoints
etcd1.example.com	etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	Compute node
node2.example.com	
infra-node1.example.com	Dedicated infrastructure node
infra-node2.example.com	

You can see these example hosts present in the **[masters]**, **[etcd]**, **[lb]**, and **[nodes]** sections of the following example inventory file:

Multiple Masters Using HAProxy Inventory File

```

# Create an OSEv3 group that contains the master, nodes, etcd, and lb
groups.
# The lb group lets Ansible configure HAProxy as the load balancing
solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
oreg_url=example.com/openshift3/ose-${component}:${version}
oreg_auth_user=${user_name}
oreg_auth_password=${password}
openshift_examples_modify_imagestreams=true

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login':
'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# apply updated node defaults
openshift_node_groups=[{'name': 'node-config-all-in-one', 'labels':
['node-role.kubernetes.io/master=true', 'node-
role.kubernetes.io/infra=true', 'node-role.kubernetes.io/compute=true'],
'edits': [{ 'key': 'kubeletArguments.pods-per-core', 'value': ['20']}]}]

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

```

```
# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```



IMPORTANT

See [Configuring Node Host Labels](#) to ensure you understand the default node selector requirements and node label considerations beginning in OpenShift Container Platform 3.9.

To use this example, modify the file to match your environment and specifications, and save it as */etc/ansible/hosts*.

5.3.2. Multiple Masters Using Native HA with Co-located Clustered etcd

The following describes an example environment for three [masters](#) using the **native** HA method (with [etcd](#) running as a static pod on each host), one HAProxy load balancer, two [nodes](#) for hosting user applications, and two nodes with the **node-role.kubernetes.io/infra=true** label for hosting [dedicated infrastructure](#):

Host Name	Component/Role(s) to Install
master1.example.com	Master (clustered using native HA) and node with etcd running as a static pod on each host
master2.example.com	
master3.example.com	
lb.example.com	HAProxy to load balance API master endpoints
node1.example.com	Compute node
node2.example.com	
infra-node1.example.com	Dedicated infrastructure node
infra-node2.example.com	

You can see these example hosts present in the **[masters]**, **[etcd]**, **[lb]**, and **[nodes]** sections of the following example inventory file:

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb
groups.
# The lb group lets Ansible configure HAProxy as the load balancing
solution.
```

```

# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
oreg_url=example.com/openshift3/ose-${component}:${version}
oreg_auth_user=${user_name}
oreg_auth_password=${password}
openshift_examples_modify_imagestreams=true

# uncomment the following to enable htpasswd authentication; defaults to
# DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login':
# 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
master1.example.com
master2.example.com
master3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'

```



IMPORTANT

See [Configuring Node Host Labels](#) to ensure you understand the default node selector requirements and node label considerations beginning in OpenShift Container Platform 3.9.

To use this example, modify the file to match your environment and specifications, and save it as ***/etc/ansible/hosts***.

CHAPTER 6. INSTALLING OPENSIFT CONTAINER PLATFORM

To install a OpenShift Container Platform cluster, you run a series of Ansible playbooks.



NOTE

To install OpenShift Container Platform as a stand-alone registry, see [Installing a Stand-alone Registry](#).

6.1. PREREQUISITES

Before installing OpenShift Container Platform, prepare your cluster hosts:

- Review the [System and environment requirements](#).
- If you will have a large cluster, review the [Scaling and Performance Guide](#) for suggestions for optimizing installation time.
- [Prepare your hosts](#). This process includes verifying system and environment requirements per component type, installing and configuring the **docker** service, and installing Ansible version 2.6 or later. You must install Ansible to run the installation playbooks.
- [Configure your inventory file](#) to define your environment and OpenShift Container Platform cluster configuration. Both your initial installation and future cluster upgrades are based on this inventory file.
- If you are installing OpenShift Container Platform on Red Hat Enterprise Linux, decide if you want to use the [RPM or system container](#) installation method. The system container method is required for RHEL Atomic Host systems.

6.1.1. Running the RPM-based installer

The RPM-based installer uses Ansible installed via RPM packages to run playbooks and configuration files available on the local host.



IMPORTANT

Do not run OpenShift Ansible playbooks under **nohup**. Using **nohup** with the playbooks causes file descriptors to be created but not closed. Therefore, the system can run out of files to open and the playbook fails.

To run the RPM-based installer:

1. Change to the playbook directory and run the **prerequisites.yml** playbook. This playbook installs required software packages, if any, and modifies the container runtimes. Unless you need to configure the container runtimes, run this playbook only once, before you deploy a cluster the first time:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
  playbooks/prerequisites.yml
```

1. If your inventory file is not in the `/etc/ansible/hosts` directory, specify `-i` and the path to the inventory file.
2. Change to the playbook directory and run the `deploy_cluster.yml` playbook to initiate the cluster installation:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \
  playbooks/deploy_cluster.yml
```

If your inventory file is not in the `/etc/ansible/hosts` directory, specify `-i` and the path to the inventory file.

. If your installation succeeded, [verify the installation](#). If your installation failed, [retry the installation](#).

6.1.2. Running the containerized installer

The `openshift3/ose-ansible` image is a containerized version of the OpenShift Container Platform installer. This installer image provides the same functionality as the RPM-based installer, but it runs in a containerized environment that provides all of its dependencies rather than being installed directly on the host. The only requirement to use it is the ability to run a container.

6.1.2.1. Running the installer as a system container

The installer image can be used as a [system container](#). System containers are stored and run outside of the traditional `docker` service. This enables running the installer image from one of the target hosts without concern for the install restarting `docker` on the host.

To use the Atomic CLI to run the installer as a run-once system container, perform the following steps as the `root` user:

1. Run the `prerequisites.yml` playbook:

```
# atomic install --system \
  --storage=ostree \
  --set INVENTORY_FILE=/path/to/inventory \
  --set PLAYBOOK_FILE=/usr/share/ansible/openshift-
ansible/playbooks/prerequisites.yml \
  --set OPTS="-v" \
  registry.redhat.io/openshift3/ose-ansible:v3.11
```

1. Specify the location on the local host for your inventory file.

This command runs a set of prerequisite tasks by using the inventory file specified and the `root` user's SSH configuration.

2. Run the `deploy_cluster.yml` playbook:

```
# atomic install --system \
  --storage=ostree \
  --set INVENTORY_FILE=/path/to/inventory \
  --set PLAYBOOK_FILE=/usr/share/ansible/openshift-
```

```
ansible/playbooks/deploy_cluster.yml \
  --set OPTS="-v" \
  registry.redhat.io/openshift3/ose-ansible:v3.11
```

- 1 Specify the location on the local host for your inventory file.

This command initiates the cluster installation by using the inventory file specified and the **root** user's SSH configuration. It logs the output on the terminal and also saves it in the `/var/log/ansible.log` file. The first time this command is run, the image is imported into **OSTree** storage (system containers use this rather than **docker** daemon storage). On subsequent runs, it reuses the stored image.

If for any reason the installation fails, before re-running the installer, see [Known Issues](#) to check for any specific instructions or workarounds.

6.1.2.2. Running other playbooks

You can use the **PLAYBOOK_FILE** environment variable to specify other playbooks you want to run by using the containerized installer. The default value of the **PLAYBOOK_FILE** is `/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml`, which is the main cluster installation playbook, but you can set it to the path of another playbook inside the container.

For example, to run the [pre-install checks](#) playbook before installation, use the following command:

```
# atomic install --system \
  --storage=ostree \
  --set INVENTORY_FILE=/path/to/inventory \
  --set PLAYBOOK_FILE=/usr/share/ansible/openshift-
ansible/playbooks/openshift-checks/pre-install.yml \ 1
  --set OPTS="-v" \ 2
  registry.redhat.io/openshift3/ose-ansible:v3.11
```

- 1 Set **PLAYBOOK_FILE** to the full path of the playbook starting at the **playbooks/** directory. Playbooks are located in the same locations as with the RPM-based installer.
- 2 Set **OPTS** to add command line options to **ansible-playbook**.

6.1.2.3. Running the installer as a Docker container

The installer image can also run as a **docker** container anywhere that **docker** can run.



WARNING

This method must not be used to run the installer on one of the hosts being configured, because the installer might restart **docker** on the host and disrupt the installation.



NOTE

Although this method and the system container method above use the same image, they run with different entry points and contexts, so runtime parameters are not the same.

At a minimum, when running the installer as a **docker** container you must provide:

- SSH key(s), so that Ansible can reach your hosts.
- An Ansible inventory file.
- The location of the Ansible playbook to run against that inventory.

Here is an example of how to run an install via **docker**, which must be run by a non-**root** user with access to **docker**:

1. First, run the *prerequisites.yml* playbook:

```
$ docker run -t -u `id -u` \ ❶
  -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \ ❷
  -v $HOME/ansible/hosts:/tmp/inventory:Z \ ❸
  -e INVENTORY_FILE=/tmp/inventory \ ❹
  -e PLAYBOOK_FILE=playbooks/prerequisites.yml \ ❺
  -e OPTS="-v" \ ❻
  registry.redhat.io/openshift3/ose-ansible:v3.11
```

❶ **-u `id -u`** makes the container run with the same UID as the current user, which allows that user to use the SSH key inside the container. SSH private keys are expected to be readable only by their owner.

❷ **-v \$HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z** mounts your SSH key, **\$HOME/.ssh/id_rsa**, under the container user's **\$HOME/.ssh** directory. **/opt/app-root/src** is the **\$HOME** of the user in the container. If you mount the SSH key into a different location, add an environment variable with **-e ANSIBLE_PRIVATE_KEY_FILE=/the/mount/point** or set **ansible_ssh_private_key_file=/the/mount/point** as a variable in the inventory to point Ansible to it. Note that the SSH key is mounted with the **:Z** flag. This flag is required so that the container can read the SSH key under its restricted SELinux context. This also means that your original SSH key file will be re-labeled to something like **system_u:object_r:container_file_t:s0:c113,c247**. For more details about **:Z**, review the **docker-run(1)** man page. Keep this in mind when providing these volume mount specifications because this might have unexpected consequences. For example, if you mount (and therefore re-label) your whole **\$HOME/.ssh** directory, it will block the host's **sshd** from accessing your public keys to log in. For this reason, you might want to use a separate copy of the SSH key or directory so that the original file labels remain untouched.

❸ ❹ **-v \$HOME/ansible/hosts:/tmp/inventory:Z** and **-e INVENTORY_FILE=/tmp/inventory** mount a static Ansible inventory file into the container as **/tmp/inventory** and set the corresponding environment variable to point at it. As with the SSH key, the inventory file SELinux labels might need to be relabeled by using the **:Z** flag to allow reading in the container, depending on the existing label. For files in a user **\$HOME** directory, this is likely to be needed. You might prefer to copy the inventory to a dedicated location before you mount it. You can also download the inventory file from a

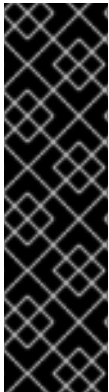
web server if you specify the **INVENTORY_URL** environment variable or generate it dynamically by using the **DYNAMIC_SCRIPT_URL** parameter to specify an executable script that provides a dynamic inventory.

- 5 **-e PLAYBOOK_FILE=playbooks/prerequisites.yml** specifies the playbook to run as a relative path from the top level directory of **openshift-ansible** content. In this example, you specify the prerequisites playbook. You can also specify the full path from the RPM or the path to any other playbook file in the container.
- 6 **-e OPTS="-v"** supplies arbitrary command line options to the **ansible-playbook** command that runs inside the container. In this example, specify **-v** to increase verbosity.

2. Next, run the **deploy_cluster.yml** playbook to initiate the cluster installation:

```
$ docker run -t -u `id -u` \
  -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \
  -v $HOME/ansible/hosts:/tmp/inventory:Z \
  -e INVENTORY_FILE=/tmp/inventory \
  -e PLAYBOOK_FILE=playbooks/deploy_cluster.yml \
  -e OPTS="-v" \
  registry.redhat.io/openshift3/ose-ansible:v3.11
```

6.1.2.4. Running the Installation Playbook for OpenStack



IMPORTANT

The OpenStack installation playbook is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for 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 on Red Hat Technology Preview features support scope, see <https://access.redhat.com/support/offerings/techpreview/>.

To install OpenShift Container Platform on an existing OpenStack installation, use the OpenStack playbook. For more information about the playbook, including detailed prerequisites, see [the OpenStack Provisioning readme file](#).

To run the playbook, run the following command:

```
$ ansible-playbook --user openshift \
  -i openshift-ansible/playbooks/openstack/inventory.py \
  -i inventory \
  openshift-ansible/playbooks/openstack/openshift-
  cluster/provision_install.yml
```

6.1.3. About the installation playbooks

The installer uses modularized playbooks so that administrators can install specific components as needed. By breaking up the roles and playbooks, there is better targeting of ad hoc administration tasks. This results in an increased level of control during installations and results in time savings.

The main installation playbook `/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml` runs a set of individual component playbooks in a specific order, and the installer reports back at the end what phases you have gone through. If the installation fails, you are notified which phase failed along with the errors from the Ansible run.



IMPORTANT

While RHEL Atomic Host is supported for running OpenShift Container Platform services as system container, the installation method uses Ansible, which is not available in RHEL Atomic Host. The RPM-based installer must therefore be run from a RHEL 7 system. The host initiating the installation does not need to be intended for inclusion in the OpenShift Container Platform cluster, but it can be. Alternatively, a [containerized version of the installer](#) is available as a system container, which can be run from a RHEL Atomic Host system.

6.2. RETRYING THE INSTALLATION

If the Ansible installer fails, you can still install OpenShift Container Platform:

1. Review the [Known Issues](#) to check for any specific instructions or workarounds.
2. Address the errors from your installation.
3. Determine if you need to uninstall and reinstall or retry the installation:
 - If you did not modify the SDN configuration or generate new certificates, retry the installation.
 - If you modified the SDN configuration, generated new certificates, or the installer fails again, you must either start over with a clean operating system installation or [uninstall](#) and install again.
 - If you use virtual machines, start from a new image or [uninstall](#) and install again.
 - If you use bare metal machines, [uninstall](#) and install again.
4. Retry the installation:
 - You can run the **`deploy_cluster.yml`** playbook again.
 - You can run the remaining individual installation playbooks.
If you want to run only the remaining playbooks, start by running the playbook for the phase that failed and then run each of the remaining playbooks in order. Run each playbook with the following command:

```
# ansible-playbook [-i /path/to/inventory]
<playbook_file_location>
```

The following table lists the playbooks in the order that they must run:

Table 6.1. Individual Component Playbook Run Order

Playbook Name	File Location
Health Check	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-checks/pre-install.yml</i>
Node Bootstrap	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-node/bootstrap.yml</i>
etcd Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-etcd/config.yml</i>
NFS Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-nfs/config.yml</i>
Load Balancer Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-loadbalancer/config.yml</i>
Master Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-master/config.yml</i>
Master Additional Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-master/additional_config.yml</i>
Node Join	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-node/join.yml</i>
GlusterFS Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-glusterfs/config.yml</i>
Hosted Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-hosted/config.yml</i>
Monitoring Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-monitoring/config.yml</i>
Web Console Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-web-console/config.yml</i>
Metrics Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml</i>
Logging Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml</i>
Prometheus Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-prometheus/config.yml</i>
Availability Monitoring Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-monitor-availability/config.yml</i>

Playbook Name	File Location
Service Catalog Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-service-catalog/config.yml</i>
Management Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-management/config.yml</i>
Descheduler Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-descheduler/config.yml</i>
Node Problem Detector Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-node-problem-detector/config.yml</i>
Autoheal Install	<i>/usr/share/ansible/openshift-ansible/playbooks/openshift-autoheal/config.yml</i>
Operator Lifecycle Manager (OLM) Install (Technology Preview)	<i>/usr/share/ansible/openshift-ansible/playbooks/olm/config.yml</i>

6.3. VERIFYING THE INSTALLATION

After the installation completes:

1. Verify that the master is started and nodes are registered and reporting in **Ready** status. *On the master host*, run the following command as root:

```
# oc get nodes
NAME                                STATUS    ROLES    AGE      VERSION
master.example.com                 Ready     master   7h
v1.9.1+a0ce1bc657
node1.example.com                  Ready     compute  7h
v1.9.1+a0ce1bc657
node2.example.com                  Ready     compute  7h
v1.9.1+a0ce1bc657
```

2. To verify that the web console is installed correctly, use the master host name and the web console port number to access the web console with a web browser.
For example, for a master host with a host name of **master.openshift.com** and using the default port of **8443**, the web console URL is **https://master.openshift.com:8443/console**.

Verifying Multiple etcd Hosts

If you installed multiple etcd hosts:

1. First, verify that the **etcd** package, which provides the **etcdctl** command, is installed:

```
# yum install etcd
```


2. On a master host, verify the etcd cluster health, substituting for the FQDNs of your etcd hosts in the following:

```
# etcdctl -C \
https://etcd1.example.com:2379,https://etcd2.example.com:2379,https://etcd3.example.com:2379 \
--ca-file=/etc/origin/master/master.etcd-ca.crt \
--cert-file=/etc/origin/master/master.etcd-client.crt \
--key-file=/etc/origin/master/master.etcd-client.key cluster-health
```

3. Also verify the member list is correct:

```
# etcdctl -C \
https://etcd1.example.com:2379,https://etcd2.example.com:2379,https://etcd3.example.com:2379 \
--ca-file=/etc/origin/master/master.etcd-ca.crt \
--cert-file=/etc/origin/master/master.etcd-client.crt \
--key-file=/etc/origin/master/master.etcd-client.key member list
```

Verifying Multiple Masters Using HAProxy

If you installed multiple masters using HAProxy as a load balancer, open the following URL and check HAProxy's status:

```
http://<lb_hostname>:9000 1
```

- 1** Provide the load balancer host name listed in the **[lb]** section of your inventory file.

You can verify your installation by consulting the [HAProxy Configuration documentation](#).

6.4. OPTIONALLY SECURING BUILDS

Running **docker build** is a privileged process, so the container has more access to the node than might be considered acceptable in some multi-tenant environments. If you do not trust your users, you can configure a more secure option after installation. Disable Docker builds on the cluster and require that users build images outside of the cluster. See [Securing Builds by Strategy](#) for more information about this optional process.

6.5. KNOWN ISSUES

- On failover in multiple master clusters, it is possible for the controller manager to overcorrect, which causes the system to run more pods than what was intended. However, this is a transient event and the system does correct itself over time. See <https://github.com/kubernetes/kubernetes/issues/10030> for details.
- Due to a known issue, after running the installation, if NFS volumes are provisioned for any component, the following directories might be created whether their components are being deployed to NFS volumes or not:
 - **/exports/logging-es**

- */exports/logging-es-ops/*
- */exports/metrics/*
- */exports/prometheus*
- */exports/prometheus-alertbuffer/*
- */exports/prometheus-alertmanager/*

You can delete these directories after installation, as needed.

6.6. WHAT'S NEXT?

Now that you have a working OpenShift Container Platform instance, you can:

- Deploy an [integrated container image registry](#).
- Deploy a [router](#).

CHAPTER 7. DISCONNECTED INSTALLATION

Frequently, portions of a data center might not have access to the Internet, even via proxy servers. You can still install OpenShift Container Platform in these environments, but you must download required software and images and make them available to the disconnected environment.

After the installation components are available to your node hosts, you install OpenShift Container Platform by following the standard installation steps.

After you install OpenShift Container Platform, you must make the S2I builder images that you pulled available to the cluster.

7.1. PREREQUISITES

- Review [OpenShift Container Platform's overall architecture](#) and plan your environment topology.
- Obtain a Red Hat Enterprise Linux (RHEL) 7 server that you have root access to with access to the Internet and at least 110 GB of disk space. You download the required software repositories and container images to this computer.
- Plan to maintain a webserver within your disconnected environment to serve the mirrored repositories. You copy the repositories from the Internet-connected host to this webserver, either over the network or by using physical media in disconnected deployments.
- Provide a source control repository. After installation, your nodes must access source code in a source code repository, such as Git.
When building applications in OpenShift Container Platform, your build might contain external dependencies, such as a Maven Repository or Gem files for Ruby applications.
- Provide a registry within the disconnected environment. Options include:
 - Installing a [stand alone OpenShift Container Platform registry](#).
 - Using a [Red Hat Satellite 6.1](#) server that acts as a container image registry.

7.2. OBTAINING REQUIRED SOFTWARE PACKAGES AND IMAGES

Before you install OpenShift Container Platform in your disconnected environment, obtain the required images and components and store them in your repository.

7.2.1. Obtaining OpenShift Container Platform packages

On the RHEL 7 server with an internet connection, sync the repositories:

1. To ensure that the packages are not deleted after you sync the repository, import the GPG key:

```
$ rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

2. Register the server with the Red Hat Customer Portal. You must use the credentials that are associated with the account that has access to the OpenShift Container Platform subscriptions:

```
$ subscription-manager register
```

3. Pull the latest subscription data from RHSM:

```
$ subscription-manager refresh
```

4. Attach a subscription that provides OpenShift Container Platform channels.

- a. Find an available subscription pool that provides the OpenShift Container Platform channels:

```
$ subscription-manager list --available --matches '*OpenShift*'
```

- b. Attach a pool ID for a subscription that provides OpenShift Container Platform:

```
$ subscription-manager attach --pool=<pool_id>
$ subscription-manager repos --disable="*"
$ subscription-manager repos \
  --enable="rhel-7-server-rpms" \
  --enable="rhel-7-server-extras-rpms" \
  --enable="rhel-7-server-ansible-2.6-rpms" \
  --enable="rhel-7-server-ose-3.11-rpms"
```

5. Install required packages:

```
$ sudo yum -y install yum-utils createrepo docker git
```

The **yum-utils** package provides the **reposync** utility, which lets you mirror yum repositories, and you can use the **createrepo** package to create a usable **yum** repository from a directory.

6. Make a directory to store the software in the server's storage or to a USB drive or other external device:

```
$ mkdir -p </path/to/repos>
```



IMPORTANT

If you can re-connect this server to the disconnected LAN and use it as the repository server, store the files locally. If you cannot, use USB-connected storage so you can transport the software to a repository server in your disconnected LAN.

7. Sync the packages and create the repository for each of them:

```
$ for repo in \
  rhel-7-server-rpms \
  rhel-7-server-extras-rpms \
  rhel-7-server-ansible-2.6-rpms \
  rhel-7-server-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=
  </path/to/repos> 1
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo}
2
done
```

- 1** **2** Provide the path to the directory you created.

7.2.2. Obtaining images

Pull the required container images:

1. Start the Docker daemon:

```
$ systemctl start docker
```

2. Pull all of the required OpenShift Container Platform infrastructure component images. Replace **<tag>** with the version to install. For example, specify **v3.11.16** for the latest version. You can specify a different minor version.

```
$ docker pull registry.redhat.io/openshift3/apb-base:<tag>
$ docker pull registry.redhat.io/openshift3/apb-tools:<tag>
$ docker pull registry.redhat.io/openshift3/automation-broker-apb:
<tag>
$ docker pull registry.redhat.io/openshift3/csi-attacher:<tag>
$ docker pull registry.redhat.io/openshift3/csi-driver-registrar:
<tag>
$ docker pull registry.redhat.io/openshift3/csi-livenessprobe:<tag>
$ docker pull registry.redhat.io/openshift3/csi-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/grafana:<tag>
$ docker pull registry.redhat.io/openshift3/image-inspector:<tag>
$ docker pull registry.redhat.io/openshift3/mariadb-apb:<tag>
$ docker pull registry.redhat.io/openshift3/mediawiki:<tag>
$ docker pull registry.redhat.io/openshift3/mediawiki-apb:<tag>
$ docker pull registry.redhat.io/openshift3/mysql-apb:<tag>
$ docker pull registry.redhat.io/openshift3/ose-ansible:<tag>
$ docker pull registry.redhat.io/openshift3/ose-ansible-service-
broker:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cli:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-autoscaler:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-capacity:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-monitoring-
operator:<tag>
$ docker pull registry.redhat.io/openshift3/ose-console:<tag>
$ docker pull registry.redhat.io/openshift3/ose-configmap-reloader:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-control-plane:<tag>
$ docker pull registry.redhat.io/openshift3/ose-deployer:<tag>
$ docker pull registry.redhat.io/openshift3/ose-descheduler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-docker-builder:<tag>
$ docker pull registry.redhat.io/openshift3/ose-docker-registry:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-efs-provisioner:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-dns-proxy:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-http-proxy:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-router:<tag>
$ docker pull registry.redhat.io/openshift3/ose-haproxy-router:<tag>
$ docker pull registry.redhat.io/openshift3/ose-hyperkube:<tag>
$ docker pull registry.redhat.io/openshift3/ose-hypershift:<tag>
```

```

$ docker pull registry.redhat.io/openshift3/ose-keepalived-
ipfailover:<tag>
$ docker pull registry.redhat.io/openshift3/ose-kube-rbac-proxy:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-kube-state-metrics:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-metrics-server:<tag>
$ docker pull registry.redhat.io/openshift3/ose-node:<tag>
$ docker pull registry.redhat.io/openshift3/ose-node-problem-
detector:<tag>
$ docker pull registry.redhat.io/openshift3/ose-operator-lifecycle-
manager:<tag>
$ docker pull registry.redhat.io/openshift3/ose-pod:<tag>
$ docker pull registry.redhat.io/openshift3/ose-prometheus-config-
reloader:<tag>
$ docker pull registry.redhat.io/openshift3/ose-prometheus-operator:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-recycler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-service-catalog:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-template-service-
broker:<tag>
$ docker pull registry.redhat.io/openshift3/ose-web-console:<tag>
$ docker pull registry.redhat.io/openshift3/postgresql-apb:<tag>
$ docker pull registry.redhat.io/openshift3/registry-console:<tag>
$ docker pull registry.redhat.io/openshift3/snapshot-controller:
<tag>
$ docker pull registry.redhat.io/openshift3/snapshot-provisioner:
<tag>
$ docker pull registry.redhat.io/rhel7/etcd:3.2.22

```

3. Pull all of the required OpenShift Container Platform component images for the optional components. Replace **<tag>** with the version to install. For example, specify **v3.11.16** for the latest version. You can specify a different minor version.

```

$ docker pull registry.redhat.io/openshift3/metrics-cassandra:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-
metrics:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-
openshift-agent:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-heapster:<tag>
$ docker pull registry.redhat.io/openshift3/oauth-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-curator5:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-
elasticsearch5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-eventrouter:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-fluentd:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-kibana5:
<tag>
$ docker pull registry.redhat.io/openshift3/ose-metrics-schema-
installer:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alert-buffer:

```

```

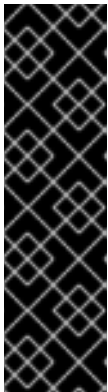
<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alertmanager:
<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-node-
exporter:<tag>
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-
postgresql
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-
memcached
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-app-ui
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-app
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-
embedded-ansible
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-httpd
$ docker pull registry.redhat.io/cloudforms46/cfme-httpd-configmap-
generator
$ docker pull registry.redhat.io/rhgs3/rhgs-server-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-volmanager-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-s3-server-rhel7

```



IMPORTANT

For Red Hat support, a converged mode subscription is required for **rhgs3/** images.



IMPORTANT

Prometheus on OpenShift Container Platform is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for 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 on Red Hat Technology Preview features support scope, see <https://access.redhat.com/support/offerings/techpreview/>.

4. Pull the Red Hat-certified [Source-to-Image \(S2I\)](#) builder images that you intend to use in your OpenShift Container Platform environment.
Make sure to indicate the correct tag by specifying the version number. See the S2I table in the [OpenShift and Atomic Platform Tested Integrations](#) page for details about image version compatibility.

You can pull the following images:

```

$ docker pull registry.redhat.io/jboss-amq-6/amq63-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datagrid-7/datagrid71-
openshift:<tag>
$ docker pull registry.redhat.io/jboss-datagrid-7/datagrid71-client-
openshift:<tag>
$ docker pull registry.redhat.io/jboss-datavirt-6/datavirt63-
openshift:<tag>
$ docker pull registry.redhat.io/jboss-datavirt-6/datavirt63-driver-
openshift:<tag>

```

```

$ docker pull registry.redhat.io/jboss-decisionserver-
6/decisionserver64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-processserver-
6/processserver64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-eap-6/eap64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-eap-7/eap70-openshift:<tag>
$ docker pull registry.redhat.io/jboss-webserver-3/webserver31-
tomcat7-openshift:<tag>
$ docker pull registry.redhat.io/jboss-webserver-3/webserver31-
tomcat8-openshift:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-2-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-agent-maven-35-
rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-agent-nodejs-8-
rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-base-
rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-maven-
rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-nodejs-
rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/mongodb-32-rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/mysql-57-rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/perl-524-rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/php-56-rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/postgresql-95-rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/python-35-rhel7:<tag>
$ docker pull registry.redhat.io/redhat-sso-7/sso70-openshift:<tag>
$ docker pull registry.redhat.io/rhsc1/ruby-24-rhel7:<tag>
$ docker pull registry.redhat.io/redhat-openjdk-18/openjdk18-
openshift:<tag>
$ docker pull registry.redhat.io/redhat-sso-7/sso71-openshift:<tag>
$ docker pull registry.redhat.io/rhsc1/nodejs-6-rhel7:<tag>
$ docker pull registry.redhat.io/rhsc1/mariadb-101-rhel7:<tag>

```

7.2.3. Exporting images

If your environment does not have access to your internal network and requires physical media to transfer content, export the images to compressed files. If your host is connected to both the Internet and your internal networks, skip the following steps and continue to [Prepare and populate the repository server](#).

1. Create a directory to store your compressed images in and change to it:

```

$ mkdir </path/to/images>
$ cd </path/to/images>

```

2. Export the OpenShift Container Platform infrastructure component images:

```

$ docker save -o ose3-images.tar \
  registry.redhat.io/openshift3/apb-base \
  registry.redhat.io/openshift3/apb-tools \
  registry.redhat.io/openshift3/automation-broker-apb \
  registry.redhat.io/openshift3/csi-attacher \
  registry.redhat.io/openshift3/csi-driver-registrar \

```



```

registry.redhat.io/openshift3/csi-livenessprobe \
registry.redhat.io/openshift3/csi-provisioner \
registry.redhat.io/openshift3/grafana \
registry.redhat.io/openshift3/image-inspector \
registry.redhat.io/openshift3/mariadb-apb \
registry.redhat.io/openshift3/mediawiki \
registry.redhat.io/openshift3/mediawiki-apb \
registry.redhat.io/openshift3/mysql-apb \
registry.redhat.io/openshift3/ose-ansible \
registry.redhat.io/openshift3/ose-ansible-service-broker \
registry.redhat.io/openshift3/ose-cli \
registry.redhat.io/openshift3/ose-cluster-autoscaler \
registry.redhat.io/openshift3/ose-cluster-capacity \
registry.redhat.io/openshift3/ose-cluster-monitoring-operator \
registry.redhat.io/openshift3/ose-console \
registry.redhat.io/openshift3/ose-configmap-reloader \
registry.redhat.io/openshift3/ose-control-plane \
registry.redhat.io/openshift3/ose-deployer \
registry.redhat.io/openshift3/ose-descheduler \
registry.redhat.io/openshift3/ose-docker-builder \
registry.redhat.io/openshift3/ose-docker-registry \
registry.redhat.io/openshift3/ose-efs-provisioner \
registry.redhat.io/openshift3/ose-egress-dns-proxy \
registry.redhat.io/openshift3/ose-egress-http-proxy \
registry.redhat.io/openshift3/ose-egress-router \
registry.redhat.io/openshift3/ose-haproxy-router \
registry.redhat.io/openshift3/ose-hyperkube \
registry.redhat.io/openshift3/ose-hypershift \
registry.redhat.io/openshift3/ose-keepalived-ipfailover \
registry.redhat.io/openshift3/ose-kube-rbac-proxy \
registry.redhat.io/openshift3/ose-kube-state-metrics \
registry.redhat.io/openshift3/ose-metrics-server \
registry.redhat.io/openshift3/ose-node \
registry.redhat.io/openshift3/ose-node-problem-detector \
registry.redhat.io/openshift3/ose-operator-lifecycle-manager \
registry.redhat.io/openshift3/ose-pod \
registry.redhat.io/openshift3/ose-prometheus-config-reloader \
registry.redhat.io/openshift3/ose-prometheus-operator \
registry.redhat.io/openshift3/ose-recycler \
registry.redhat.io/openshift3/ose-service-catalog \
registry.redhat.io/openshift3/ose-template-service-broker \
registry.redhat.io/openshift3/ose-web-console \
registry.redhat.io/openshift3/postgresql-apb \
registry.redhat.io/openshift3/registry-console \
registry.redhat.io/openshift3/snapshot-controller \
registry.redhat.io/openshift3/snapshot-provisioner
registry.redhat.io/rhel7/etcd:3.2.22

```

1. If you synchronized images for optional components, export them:

```

$ docker save -o ose3-optional-imgs.tar \
  registry.redhat.io/openshift3/metrics-cassandra \
  registry.redhat.io/openshift3/metrics-hawkular-metrics \
  registry.redhat.io/openshift3/metrics-hawkular-openshift-agent \
  registry.redhat.io/openshift3/metrics-heapster \
  registry.redhat.io/openshift3/oauth-proxy \

```

```

registry.redhat.io/openshift3/ose-logging-curator5 \
registry.redhat.io/openshift3/ose-logging-elasticsearch5 \
registry.redhat.io/openshift3/ose-logging-eventrouter \
registry.redhat.io/openshift3/ose-logging-fluentd \
registry.redhat.io/openshift3/ose-logging-kibana5 \
registry.redhat.io/openshift3/ose-metrics-schema-installer \
registry.redhat.io/openshift3/prometheus \
registry.redhat.io/openshift3/prometheus-alert-buffer \
registry.redhat.io/openshift3/prometheus-alertmanager \
registry.redhat.io/openshift3/prometheus-node-exporter \
registry.redhat.io/cloudforms46/cfme-openshift-postgresql \
registry.redhat.io/cloudforms46/cfme-openshift-memcached \
registry.redhat.io/cloudforms46/cfme-openshift-app-ui \
registry.redhat.io/cloudforms46/cfme-openshift-app \
registry.redhat.io/cloudforms46/cfme-openshift-embedded-ansible \
\
registry.redhat.io/cloudforms46/cfme-openshift-httpd \
registry.redhat.io/cloudforms46/cfme-httpd-configmap-generator \
registry.redhat.io/rhgs3/rhgs-server-rhel7 \
registry.redhat.io/rhgs3/rhgs-volmanager-rhel7 \
registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7 \
registry.redhat.io/rhgs3/rhgs-s3-server-rhel7

```

2. Export the S2I builder images that you pulled. For example, if you synced only the Jenkins and Tomcat images:

```

$ docker save -o ose3-builder-images.tar \
  registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-
openshift:<tag> \
  registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-
openshift:<tag> \
  registry.redhat.io/openshift3/jenkins-2-rhel7:<tag> \
  registry.redhat.io/openshift3/jenkins-agent-maven-35-rhel7:<tag> \
\
  registry.redhat.io/openshift3/jenkins-agent-nodejs-8-rhel7:<tag> \
\
  registry.redhat.io/openshift3/jenkins-slave-base-rhel7:<tag> \
  registry.redhat.io/openshift3/jenkins-slave-maven-rhel7:<tag> \
  registry.redhat.io/openshift3/jenkins-slave-nodejs-rhel7:<tag>

```

3. Copy the compressed files from your Internet-connected host to your internal host.
4. Load the images that you copied:

```

$ docker load -i ose3-images.tar
$ docker load -i ose3-builder-images.tar
$ docker load -i ose3-optional-images.tar

```

7.3. PREPARE AND POPULATE THE REPOSITORY SERVER

During the installation, and any future updates, you need a webserver to host the software. RHEL 7 can provide the Apache webserver.

1. Prepare the webserver:

- a. If you need to install a new webserver in your disconnected environment, install a new RHEL 7 system with at least 110 GB of space on your LAN. During RHEL installation, select the **Basic Web Server** option.
- b. If you are re-using the server where you downloaded the OpenShift Container Platform software and required images, install Apache on the server:

```
$ sudo yum install httpd
```

2. Place the repository files into Apache's root folder.

- If you are re-using the server:

```
$ mv /path/to/repos /var/www/html/
$ chmod -R +r /var/www/html/repos
$ restorecon -vR /var/www/html
```

- If you installed a new server, attach external storage and then copy the files:

```
$ cp -a /path/to/repos /var/www/html/
$ chmod -R +r /var/www/html/repos
$ restorecon -vR /var/www/html
```

3. Add the firewall rules:

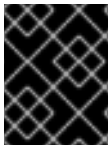
```
$ sudo firewall-cmd --permanent --add-service=http
$ sudo firewall-cmd --reload
```

4. Enable and start Apache for the changes to take effect:

```
$ systemctl enable httpd
$ systemctl start httpd
```

7.4. POPULATE THE REGISTRY

From within your disconnected environment, tag and push the images to your internal registry:



IMPORTANT

The following steps are a generic guide to loading the images into a registry. You might need to take more or different actions to load the images.

1. Before you push the images into the registry, re-tag each image.
 - For images in the **openshift3** repository, tag the image as both the major and minor version number. For example, to tag the OpenShift Container Platform node image:

```
$ docker tag registry.redhat.io/openshift3/ose-node:<tag>
registry.example.com/openshift3/ose-node:<tag>
$ docker tag registry.redhat.io/openshift3/ose-node:<tag>
registry.example.com/openshift3/ose-node:{major-tag}
```

- For other images, tag the image with the exact version number. For example, to tag the etcd image:

```
$ docker tag registry.redhat.io/rhel7/etcd:3.2.22
registry.example.com/rhel7/etcd:3.2.22
```

2. Push each image into the registry. For example, to push the OpenShift Container Platform node images:

```
$ docker push registry.example.com/openshift3/ose-node:<tag>
$ docker push registry.example.com/openshift3/ose-node:{major-tag}
```

7.5. PREPARING CLUSTER HOSTS

Now that you have the installation files, prepare your hosts.

1. Create the hosts for your OpenShift Container Platform cluster. It is recommended to use the latest version of RHEL 7 and to perform a minimal installation. Ensure that the hosts meet the [system requirements](#).
2. On each node host, create the repository definitions. Place the following text in the `/etc/yum.repos.d/ose.repo` file:

```
[rhel-7-server-rpms]
name=rhel-7-server-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-rpms ❶
enabled=1
gpgcheck=0
[rhel-7-server-extras-rpms]
name=rhel-7-server-extras-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-extras-rpms ❷
enabled=1
gpgcheck=0
[rhel-7-server-ansible-2.6-rpms]
name=rhel-7-server-ansible-2.6-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ansible-2.6-rpms ❸
enabled=1
gpgcheck=0
[rhel-7-server-ose-3.11-rpms]
name=rhel-7-server-ose-3.11-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ose-3.11-rpms ❹
enabled=1
gpgcheck=0
```

- ❶ ❷ ❸ ❹ Replace `<server_IP>` with the IP address or host name of the Apache server that hosts the software repositories.

3. Finish preparing the hosts for installation. Follow the [Preparing your hosts](#) steps, omitting the steps in the **Host Registration** section.

7.6. INSTALLING OPENSIFT CONTAINER PLATFORM

After you prepare the software, images, and hosts, you use the standard installation method to install OpenShift Container Platform:

1. [Configure your inventory file](#) to reference your internal registry:

```
orge_url=registry.example.com/openshift3/ose-${component}:${version}
openshift_examples_modify_imagestreams=true
```

2. [Run the installation playbooks](#).

CHAPTER 8. INSTALLING A STAND-ALONE DEPLOYMENT OF OPENSIFT CONTAINER IMAGE REGISTRY

OpenShift Container Platform is a fully-featured enterprise solution that includes an integrated container image registry called [OpenShift Container Registry](#) (OCR). Instead of deploying OpenShift Container Platform as a full Platform-as-a-Service environment for developers, you can install OCR as a stand-alone container image registry to run on-site or in the cloud.

When installing a stand-alone deployment of OCR, a cluster of masters and nodes is still installed, similar to a typical OpenShift Container Platform installation. Then, the container image registry is deployed to run on the cluster. This stand-alone deployment option is useful for administrators that want a container image registry but do not require the full OpenShift Container Platform environment that includes the developer-focused web console and application build and deployment tools.

OCR provides the following capabilities:

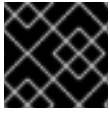
- A user-focused registry web console, [Cockpit](#).
- [Secured traffic](#) by default, served via TLS.
- Global [identity provider authentication](#).
- A [project namespace](#) model to enable teams to collaborate through [role-based access control \(RBAC\)](#) authorization.
- A [Kubernetes-based cluster](#) to manage services.
- An image abstraction called [image streams](#) to enhance image management.

Administrators can deploy a stand-alone OCR to manage a registry separately that supports multiple OpenShift Container Platform clusters. A stand-alone OCR also enables administrators to separate their registry to satisfy their own security or compliance requirements.

8.1. MINIMUM HARDWARE REQUIREMENTS

Installing a stand-alone OCR has the following hardware requirements:

- Physical or virtual system or an instance running on a public or private IaaS.
- Base OS: RHEL 7.4 or 7.5 with the "Minimal" installation option and the latest packages from the RHEL 7 Extras channel, or RHEL Atomic Host 7.4.5 or later.
- NetworkManager 1.0 or later.
- 2 vCPU.
- Minimum 16 GB RAM.
- Minimum 15 GB hard disk space for the file system containing `/var/`.
- An additional minimum 15 GB unallocated space for Docker's storage back end; see [Configuring Docker Storage](#) for details.

**IMPORTANT**

OpenShift Container Platform only supports servers with x86_64 architecture.

**NOTE**

To meet the `/var/` file system sizing requirements in RHEL Atomic Host you must modify the default configuration. See [Managing Storage in Red Hat Enterprise Linux Atomic Host](#) for instructions on configuring this during or after installation.

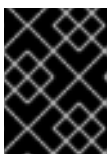
8.2. SUPPORTED SYSTEM TOPOLOGIES

The following system topologies are supported for stand-alone OCR:

All-in-one	A single host that includes the master, node, and registry components.
Multiple masters (Highly-Available)	Three hosts with all components, master, node, and registry, included on each with the masters configured for native high-availability.

8.3. INSTALLING THE OPENSIFT CONTAINER REGISTRY

1. Review the full cluster installation process, starting with [Planning Your Installation](#). Installing OCR uses the same process but requires a few specific settings in the inventory file. The installation documentation includes a comprehensive list of available Ansible variables for the inventory file.
2. Complete the [host preparation](#) steps.
3. Create an [inventory file](#) in the `/etc/ansible/hosts` directory:

**IMPORTANT**

To install a standalone OCR, you must set `deployment_subtype=registry` in the inventory file in the `[OSEv3:vars]` section.

Use the following example inventory files for the different supported system topologies:

All-in-one stand-alone OpenShift Container Registry inventory file

```
# Create an OSEv3 group that contains the masters and nodes groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring
a password
ansible_ssh_user=root
```

```

openshift_master_default_subdomain=apps.test.example.com

# If ansible_ssh_user is not root, ansible_become must be set to
true
#ansible_become=true

openshift_deployment_type=openshift-enterprise
deployment_subtype=registry ❶
openshift_hosted_infra_selector="" ❷

# uncomment the following to enable htpasswd authentication;
defaults to DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth',
'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
registry.example.com

# host group for etcd
[etcd]
registry.example.com

# host group for nodes
[nodes]
registry.example.com openshift_node_group_name='node-config-all-in-
one'

```

- ❶ Set **deployment_subtype=registry** to ensure installation of stand-alone OCR and not a full OpenShift Container Platform environment.
- ❷ Allows the registry and its web console to be scheduled on the single host.

Multiple masters (highly-available) stand-alone OpenShift Container Registry inventory file

```

# Create an OSEv3 group that contains the master, nodes, etcd, and
lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing
solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
deployment_subtype=registry ❶

```



```

openshift_master_default_subdomain=apps.test.example.com

# Uncomment the following to enable htpasswd authentication;
defaults to
# DenyAllPasswordIdentityProvider.
#openshift_master_identity_providers=[{'name': 'htpasswd_auth',
'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]]

# Native high availability cluster method with optional load
balancer.
# If no lb group is defined installer assumes that a load balancer
has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load
balancer
# or to one or all of the masters defined in the inventory if no
load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-
cluster.example.com

# apply updated node-config-compute group defaults
openshift_node_groups=[{'name': 'node-config-compute', 'labels':
['node-role.kubernetes.io/compute=true'], 'edits': [{'key':
'kubeletArguments.pods-per-core', 'value': ['20']}, {'key':
'kubeletArguments.max-pods', 'value': ['250']}, {'key':
'kubeletArguments.image-gc-high-threshold', 'value': ['90']}, {'key':
'kubeletArguments.image-gc-low-threshold', 'value': ['80']}]}}]

# enable ntp on masters to ensure proper failover
openshift_clock_enabled=true

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

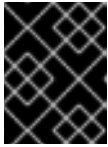
# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-
master-infra'

```

```
node1.example.com    openshift_node_group_name='node-config-
compute'
node2.example.com    openshift_node_group_name='node-config-
compute'
```

- 1 Set **deployment_subtype=registry** to ensure installation of stand-alone OCR and not a full OpenShift Container Platform environment.

4. Install the stand-alone OCR. The process is similar to a full [cluster installation](#) process.



IMPORTANT

The host that you run the Ansible playbook on must have at least 75MiB of free memory per host in the inventory file.

- a. Before you deploy a new cluster, change to the cluster directory and run the ***prerequisites.yml*** playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
  playbooks/prerequisites.yml
```

- 1 If your inventory file is not in the ***/etc/ansible/hosts*** directory, specify ***-i*** and the path to the inventory file.

You must run this playbook only one time.

- b. To initiate installation, change to the playbook directory and run the ***deploy_cluster.yml*** playbook:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
  playbooks/deploy_cluster.yml
```

- 1 If your inventory file is not in the ***/etc/ansible/hosts*** directory, specify ***-i*** and the path to the inventory file.

CHAPTER 9. UNINSTALLING OPENSIFT CONTAINER PLATFORM

You can uninstall OpenShift Container Platform hosts in your cluster by running the ***uninstall.yml*** playbook. This playbook deletes OpenShift Container Platform content installed by Ansible, including:

- Configuration
- Containers
- Default templates and image streams
- Images
- RPM packages

The playbook deletes content for any hosts defined in the inventory file that you specify when running the playbook.

CHAPTER 10. UNINSTALLING A OPENSIFT CONTAINER PLATFORM CLUSTER

To uninstall OpenShift Container Platform across all hosts in your cluster, change to the playbook directory and run the playbook using the inventory file you used most recently:

```
# ansible-playbook [-i /path/to/file] \ 1
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

- 1** If your inventory file is not in the `/etc/ansible/hosts` directory, specify `-i` and the path to the inventory file.

10.1. UNINSTALLING NODES

To uninstall node components from specific hosts using the ***uninstall.yml*** playbook while leaving the remaining hosts and cluster alone:



WARNING

Use this method only when attempting to uninstall specific node hosts, not specific masters or etcd hosts. Uninstalling master or etcd hosts requires more configuration changes in the cluster.

1. Follow the steps in [Deleting Nodes](#) to remove the node object from the cluster.
2. Create a different inventory file that references only those hosts. For example, to delete content from only one node:

```
[OSEv3:children]
nodes 1

[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

[nodes]
node3.example.com openshift_node_group_name='node-config-infra' 2
```

- 1** Include only the sections that apply to the hosts to uninstall.
- 2** Include only the hosts to uninstall.

3. Change to the playbook directory and run the ***uninstall.yml*** playbook:

```
# ansible-playbook -i /path/to/new/file \ ❶  
    /usr/share/ansible/openshift-  
    ansible/playbooks/adhoc/uninstall.yml
```

- ❶ Specify the path to the new inventory file.

When the playbook completes, all OpenShift Container Platform content is removed from the specified hosts.