OpenShift Container Platform 3.10

Installing Clusters

OpenShift Container Platform 3.10 Installing Clusters

Last Updated: 2019-11-05

OpenShift Container Platform 3.10 Installing Clusters

OpenShift Container Platform 3.10 Installing Clusters

Legal Notice

Abstract

Install your OpenShift Container Platform 3.10 cluster with this guide

Table of Contents

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:

- *Do your on-premise servers use IBM POWER or x86_64 processors?* You can install OpenShift Container Platform on servers that use either type of processor. If you use POWER servers, review the Limitations and Considerations for Installations on IBM POWER .

- *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.

- *Do 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.

- *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.

1.1.1. Limitations and Considerations for Installations on IBM POWER

As of version 3.10.45, you can install OpenShift Container Platform on IBM POWER servers.

- Your cluster must use only Power nodes and masters. Because of the way that images are tagged, OpenShift Container Platform cannot differentiate between x86 images and Power images.

- Image streams and templates are not installed by default or updated when you upgrade. You can manually install and update the image streams.

- You can install only on on-premise Power servers. You cannot install OpenShift Container Platform on nodes in any cloud provider.

- Not all storage providers are supported. You can use only the following storage providers:

  - GlusterFS
  - NFS
  - Local storage

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**
- **openshift3/node**
- **openshift3/openvswitch**
- **registry.access.redhat.com/rhel7/etcd**

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

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 Docker 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 following sections identify the hardware specifications and system-level requirements of all hosts within your OpenShift Container Platform environment.

2.1.1. Red Hat Subscriptions

You must have an active OpenShift Container Platform subscription on your Red Hat account to proceed. 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	● Physical or virtual system, or an instance running on a public or private IaaS. ● Base OS: Red Hat Enterprise Linux (RHEL) 7.4 or laterwith the "Minimal" installation option and the latest packages from the Extras channel, or RHEL Atomic Host 7.4.5 or later. ○ IBM POWER9: RHEL-ALT 7.5 with the "Minimal" installation option and the latest packages from the Extras channel. ○ IBM POWER8: RHEL 7.5 with the "Minimal" installation option and the latest packages from the Extras channel. ● 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 ***/var/***. 1 ● Minimum 1 GB hard disk space for the file system containing ***/usr/local/bin/***. ● Minimum 1 GB hard disk space for the file system containing the system's temporary directory. 2 ● Masters with a co-located etcd require a minimum of 4 cores. 2 core systems will not work.

Nodes	● Physical or virtual system, or an instance running on a public or private IaaS. ● Base OS: RHEL 7.4 or laterwith "Minimal" installation option, orRHEL Atomic Host 7.4.5 or later. ○ IBM POWER9: RHEL-ALT 7.5 with the "Minimal" installation option and the latest packages from the Extras channel. ○ IBM POWER8: RHEL 7.5 with the "Minimal" installation option and the latest packages from the Extras channel. ● NetworkManager 1.0 or later. ● 1 vCPU. ● Minimum 8 GB RAM. ● Minimum 15 GB hard disk space for the file system containing */var/*. [1] ● Minimum 1 GB hard disk space for the file system containing */usr/local/bin/*. ● 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	● Minimum 20 GB hard disk space for etcd data. ● See the Hardware Recommendations section of the CoreOS etcd documentationfor 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.

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 is run in containers on masters, and storage is needed on masters to run the web console. Containers are run on nodes, so storage is always required on the nodes. The size of storage depends on workload, number of containers, the size of the containers being run, and the containers' storage requirements. Containerized etcd also needs container storage configured.

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 should have, in addition to the minimum requirements in the table above, 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 would be 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.

A minimum of three etcd hosts and a load-balancer between the master hosts are required.

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 will need to calculate the expected workload, then 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. Storage management

Table 2.1. The main directories to which OpenShift Container Platform components write data

Directory	Notes	Sizing	Expected Growth
/var/lib/openshift	Used for etcd storage only when in single master mode and etcd is embedded in the **atomic-openshift-master** process.	Less than 10GB.	Will grow slowly with the environment. Only storing metadata.
/var/lib/etcd	Used for etcd storage when in Multi-Master mode or when etcd is made standalone by an administrator.	Less than 20 GB.	Will grow slowly with the environment. Only storing metadata.

Directory	Notes	Sizing	Expected Growth
/var/lib/docker	When the run time is docker, this is the mount point. Storage used for active container runtimes (including pods) and storage of local images (not used for registry storage). Mount point should be managed by docker-storage rather than manually.	50 GB for a Node with 16 GB memory. Additional 20-25 GB for every additional 8 GB of memory.	Growth is limited by the capacity for running containers.
/var/lib/containers	When the run time is CRI-O, this is the mount point. Storage used for active container runtimes (including pods) and storage of local images (not used for registry storage).	50 GB for a Node with 16 GB memory. Additional 20-25 GB for every additional 8 GB of memory.	Growth limited by capacity for running containers
/var/lib/origin/openshift.local.volumes	Ephemeral volume storage for pods. This includes anything external that is mounted into a container at runtime. Includes environment variables, kube secrets, and data volumes not backed by persistent storage PVs.	Varies	Minimal if pods requiring storage are using persistent volumes. If using ephemeral storage, this can grow quickly.
/var/log	Log files for all components.	10 to 30 GB.	Log files can grow quickly; size can be managed by growing disks or managed using log rotate.

2.1.5. 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.

2.1.6. 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 should 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 would need to 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 could 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
```

> **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 may fail to resolve host names properly.

2.2.1.3. Configuring Node Host Names

When you set up a cluster that is not integrated with a cloud provider, you must correctly set your nodes' host names. Each node's host name must be resolvable, and each node must be able to reach each other node.

To confirm that a node can reach another node:

1. On one node, obtain the host name:

   ```
   $ hostname

   master-1.example.com
   ```

2. On that same node, obtain the fully qualified domain name of the host:

   ```
   $ hostname -f

   master-1.example.com
   ```

3. From a different node, confirm that you can reach the first node:

   ```
   $ ping master-1.example.com -c 1

   PING master-1.example.com (172.16.122.9) 56(84) bytes of data.
   64 bytes from master-1.example.com (172.16.122.9): icmp_seq=1 ttl=64 time=0.319 ms

   --- master-1.example.com ping statistics ---
   1 packets transmitted, 1 received, 0% packet loss, time 0ms
   rtt min/avg/max/mdev = 0.319/0.319/0.319/0.000 ms
   ```

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 should 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 would need to 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 2.2. Node to Node

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

Table 2.3. 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** may 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 2.4. 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. This port must to be allowed from masters and infra nodes to any master and node. For metrics, the source must be the infra nodes.
10010	TCP	If using CRI-O, open this port to allow **oc exec** and **oc rsh** operations.

Table 2.5. 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** may 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 2.6. External to Load Balancer

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

Table 2.7. 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 2.8. 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** may 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 2.9. Aggregated Logging and Metrics

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 open 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 can communicate. Add required ports for Prometheus to required ports section with each other.
9090	TCP	For Prometheus API and web console use.
9100	TCP	For the Prometheus Node-Exporter, which exports hardware and operating system metrics. Port 9100 needs to be open on each OpenShift Container Platform host in order for the Prometheus server to scrape the metrics.
8443	TCP	For node hosts to communicate to the master API, for the node hosts to post back status, to receive tasks, and so on. This port needs to be allowed from masters and infra nodes to any master and node.
10250	TCP	For the Kubernetes cAdvisor, a container resource usage and performance analysis agent. This port must to be allowed from masters and infra nodes to any master and node. For metrics, the source must be the infra nodes.
8444	TCP	Port that the controller service listens on. Port 8444 needs to be open on each OpenShift Container Platform host

1936	TCP	(**Optional**) Required to be open when running the template router to access statistics. This port must be allowed from the infra nodes to any infra nodes hosting the routers if Prometheus metrics are enabled on routers. 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 above for more information.

Notes

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, run the **openshift_facts** playbook:

```
# ansible-playbook  [-i /path/to/inventory] \
    /usr/share/ansible/openshift-ansible/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 Configuring Your Inventory File topic discusses the available Ansible variables in greater detail.

Variable	Usage
hostname	- Resolves to the internal IP address from the instances themselves.
ip	- Should be the internal IP of the instance.
public_hostname	- Should resolve to the external IP from hosts outside of the cloud. - Provider **openshift_public_hostname** overrides.
public_ip	- Should be the externally accessible IP associated with the instance. - **openshift_public_ip** overrides.
use_openshift_sdn	- Should be true unless the cloud is GCE. - **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

3.1. OPERATING SYSTEM REQUIREMENTS

The operating system requirements for master and node hosts are different depending on your server architecture.

- For servers that use x86_64 architecture, use a base installation of Red Hat Enterprise Linux (RHEL) 7.4 or later with the latest packages from the Extras channel or RHEL Atomic Host 7.4.2 or later.
- For cloud-based installations, use a base installation of RHEL 7.4 or later with the latest packages from the Extras channel.
- For servers that use IBM POWER8 architecture, use a base installation of RHEL 7.5 with the latest packages from the Extras channel.
- For servers that use IBM POWER9 architecture, use a base installation of RHEL-ALT 7.5 with the latest packages from the Extras channel.

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. SERVER TYPE REQUIREMENTS

If you use IBM POWER servers for your nodes, you can use only IBM POWER servers. You cannot add nodes that run on IBM POWER servers to an existing cluster that uses x86_64 servers or deploy cluster nodes on a mix of IBM POWER and x86_64 servers.

3.3. SETTING PATH

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

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

These should all be included by default in a fresh RHEL 7.x installation.

3.4. 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, passwordless **sudo** rights must be configured on each destination host.

For example, you can generate an SSH key on the host where you will invoke the installation process:

```
# ssh-keygen
```

Do **not** use a password.

An easy way to distribute your SSH keys is by using a **bash** loop:

```
# for host in 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.

Modify the host names in the above command according to your configuration.

After you run the **bash** loop, confirm that you can access each host that is listed in the loop through SSH.

3.5. 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
```

3.6. HOST REGISTRATION

Each host must be registered using Red Hat Subscription Manager (RHSM) and have an active OpenShift Container Platform subscription attached to access the required packages.

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 could take a few minutes if you have a large number of available repositories

6. Enable only the repositories required by OpenShift Container Platform 3.10.

   - For cloud installations and on-premise installations on x86_64 servers, run the following command:

     ```
     # subscription-manager repos \
         --enable="rhel-7-server-rpms" \
     ```

```
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.10-rpms" \
    --enable="rhel-7-server-ansible-2.4-rpms"
```

- For on-premise installations on IBM POWER8 servers, run the following command:

  ```
  # subscription-manager repos \
      --enable="rhel-7-for-power-le-rpms" \
      --enable="rhel-7-for-power-le-extras-rpms" \
      --enable="rhel-7-for-power-le-optional-rpms" \
      --enable="rhel-7-server-ansible-2.6-for-power-le-rpms" \
      --enable="rhel-7-server-for-power-le-rhscl-rpms" \
      --enable="rhel-7-for-power-le-ose-3.10-rpms"
  ```

- For on-premise installations on IBM POWER9 servers, run the following command:

  ```
  # subscription-manager repos \
      --enable="rhel-7-for-power-9-rpms" \
      --enable="rhel-7-for-power-9-extras-rpms" \
      --enable="rhel-7-for-power-9-optional-rpms" \
      --enable="rhel-7-server-ansible-2.6-for-power-9-rpms" \
      --enable="rhel-7-server-for-power-le-rhscl-rpms" \
      --enable="rhel-7-for-power-le-ose-3.10-rpms"
  ```

3.7. INSTALLING BASE PACKAGES

NOTE

If your hosts are running RHEL 7.5 and you want to accept OpenShift Container Platform's default **docker** configuration (using OverlayFS storage and all default logging options), you can skip to Configuring Your Inventory File to create an inventory representing your cluster. A **prerequisites.yml** playbook used when running the installation will ensure that the default packages and configuration are correctly applied.

If your hosts are running RHEL 7.4 or if they are running RHEL 7.5 and you want to customize the **docker** configuration further, following the guidance in the remaining sections of this topic.

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. If you plan to use the RPM-based installer to run the installation, you can skip this step. However, if you plan to use the containerized installer:

   a. Install the **atomic** package:

   a. Install the **atomic** package:

      ```
      # yum install atomic
      ```

   b. Skip to Installing Docker.

4. Install the following package, which provides RPM-based OpenShift Container Platform installer utilities and pulls in other packages required by the cluster installation process, such as Ansible, playbooks, and related configuration files:

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

   > NOTE
   >
   > In previous OpenShift Container Platform releases, the **atomic-openshift-utils** package was installed for this step. However, starting with OpenShift Container Platform 3.10, that package is removed and the **openshift-ansible** package provides all requirements.

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:

   ```
   # systemctl reboot
   ```

3.8. INSTALLING DOCKER

At this point, you should install Docker on all master and node hosts. This allows you to configure your Docker storage options before installing OpenShift Container Platform.

For RHEL 7 systems, install Docker 1.13:

> NOTE
>
> On RHEL Atomic Host 7 systems, Docker should already be installed, configured, and running by default.

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

After the package installation is complete, verify that version 1.13 was installed:

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

> NOTE
>
> The cluster installation process automatically modifies the ***/etc/sysconfig/docker*** file.

3.9. 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 each system that runs a container daemon. For containerized installations, you need storage on masters. Also, by default, the web console is run in containers on masters, and storage is needed on masters to run the web console. Containers are run on nodes, so storage is always required on the nodes. The size of storage depends on workload, number of containers, the size of the containers being run, and the containers' storage requirements. Containerized etcd also needs container storage configured.

NOTE

If your hosts are running RHEL 7.5 and you want to accept OpenShift Container Platform's default **docker** configuration (using OverlayFS storage and all default logging options), you can skip to Configuring Your Inventory File to create an inventory representing your cluster. A **prerequisites.yml** playbook used when running the installation will ensure that the default packages and configuration are correctly applied.

If your hosts are running RHEL 7.4 or if they are running RHEL 7.5 and you want to customize the **docker** configuration further, following the guidance in the remaining sections of this topic.

For RHEL Atomic Host

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 allocated, see Managing Storage with Docker Formatted Containers for details on using **docker-storage-setup** and basic instructions on storage management in RHEL Atomic Host.

For RHEL

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 and is not supported prior to Red Hat Enterprise Linux 7.2. See the Red Hat Enterprise Linux release notes for information on using OverlayFS with your version of RHEL.

For more information on the benefits and limitations of DeviceMapper and OverlayFS, see Choosing a Graph Driver.

3.9.1. 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 on enabling the OverlayFS storage driver for the Docker service, see the Red Hat Enterprise Linux Atomic Host documentation.

3.9.2. 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. This can be done after installing Docker and should be done before creating 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:

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

Option A is the most robust option, however it requires adding an additional block device to your host before configuring Docker storage. Options B and C both require leaving free space available when provisioning your host. Option C 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:

    - **Option A)** Use an additional block device.
      In ***/etc/sysconfig/docker-storage-setup***, set **DEVS** to the path of the block device you wish to use. Set **VG** to the volume group name you wish to create; **docker-vg** is a reasonable choice. For example:

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

      Then 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.
```

- **Option B)** Use an existing, specified volume group.
  In ***/etc/sysconfig/docker-storage-setup***, set **VG** to the desired volume group. For example:

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

  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.
  ```

- **Option C)** Use the remaining free space from the volume group where your root file system is located.
  Verify that the volume group where your root file system resides has the desired 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. You should have a **dm.thinpooldev** value in the ***/etc/sysconfig/docker-storage*** file and a **docker-pool** logical volume:

```
# 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. The **docker-pool** volume should be 60% of the available volume group and will grow to fill the volume group via LVM monitoring.

3. If Docker has not yet been started on the host, enable and start the service, then verify it is running:

```
# systemctl enable docker
# systemctl start docker
# systemctl is-active docker
```

If Docker is already running, 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
```

If there is any content in ***/var/lib/docker/***, it must be deleted. Files will be present if Docker has been used prior to the installation of OpenShift Container Platform.

3.9.3. Reconfiguring Docker Storage

Should you need to reconfigure Docker storage after having created the **docker-pool**, you should first remove the **docker-pool** logical volume. If you are using a dedicated volume group, you should also remove the volume group and any associated physical volumes before reconfiguring **docker-storage-setup** according to the instructions above.

See Logical Volume Manager Administration for more detailed information on LVM management.

3.9.4. Enabling Image Signature Support

OpenShift Container Platform is capable of cryptographically verifying 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.

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

```
$ yum install atomic
```

The **atomic trust** sub-command manages trust configuration. The default configuration is to whitelist all registries. This means no signature verification is configured.

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

A reasonable configuration might be to whitelist a particular registry or namespace, blacklist (reject) untrusted registries, and require signature verification on a vendor registry. The following set of commands performs this example configuration:

Example Atomic Trust Configuration

```
$ 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.access.redhat.com

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

```
172.30.1.1:5000                accept
172.30.1.1:5000/production         signed security@example.com
registry.access.redhat.com         signed security@redhat.com,security@redhat.com
```

When all the signed sources are verified, nodes may be further hardened with a global **reject** default:

```
$ 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.access.redhat.com         signed security@redhat.com,security@redhat.com
```

Use the **atomic** man page **man atomic-trust** for additional examples.

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

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

The trust configuration may be managed directly on each node or the generated files managed on a separate host and distributed to the appropriate nodes using Ansible, for example. See the Container Image Signing Integration Guide for an example of automating file distribution with Ansible.

3.9.5. Managing Container Logs

Sometimes a container's log file (the */var/lib/docker/containers/<hash>/<hash>-json.log* file on the node where the container is running) can increase to a problematic size. You can manage this by configuring Docker's **json-file** logging driver to restrict the size and number of log files.

Option	Purpose
--log-opt max-size	Sets the size at which a new log file is created.
--log-opt max-file	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 1MB and always keep the last three log files, append **max-size=1M** and **max-file=3** to the **OPTIONS=** line, ensuring that the values maintain the single quotation mark formatting:

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

2. Restart the Docker service:

   ```
   # systemctl restart docker
   ```

3.9.6. Viewing Available Container Logs

Container logs are stored 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
```

See Docker's documentation for additional information on how to configure logging drivers.

3.9.7. 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 could 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
```

3.10. 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 if your servers use x86_64 architecture. 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
```

3.11. WHAT'S NEXT?

After you have finished preparing your hosts, you can proceed to configure your inventory file .

NOTE

If you are installing a stand-alone registry, continue instead with the Installing a Stand-alone Registry topic.

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, as well as 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 on the format of an inventory file, including basics on 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 desired configuration per 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. Accepting the default values for required variables should suffice for development environments, but for production environments, it is recommended you read through and become familiar with the various options available.

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 environment variables during the Ansible install that apply more globally to your OpenShift Container Platform cluster overall, indicate the desired variables in the ***/etc/ansible/hosts*** file on separate, single lines within the **[OSEv3:vars]** section. 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 variables for use with the Ansible installer that can be assigned cluster-wide:

Table 4.1. General Cluster Variables

Variable	Purpose
ansible_ssh_user	This variable sets the SSH user for the installer to use and defaults to **root**. This user should allow SSH-based authentication without requiring a password. If using SSH key-based authentication, then the key should be managed by an SSH agent. IMPORTANT If you plan to deploy using the **deploy_cluster.yml** playbook, you should define the **ansible_ssh_user** variable.
ansible_become	If **ansible_ssh_user** is not **root**, this variable must be set to **true** and the user must be configured for passwordless **sudo**.
debug_level	This variable sets which INFO messages are logged to the **systemd-journald.service**. Set one of the following: ● **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 For more information on debug log levels, see Configuring Logging Levels.

Variable	Purpose
openshift_clock_enabled	Whether to enable Network Time Protocol (NTP) on cluster nodes. The default value is **true**. If the **chrony** package is installed, it is configured to provide NTP service. If the **chrony** package is not installed, the installation playbooks install and configure the **ntp** package to provide NTP service. IMPORTANT To prevent masters and nodes in the cluster from going out of sync, do not change the default value of this parameter.
openshift_master_admission_plugin_config	This variable sets the parameter and arbitrary JSON values as per the requirement in your inventory hosts file. For example: `openshift_master_admission_plugin_config={"ClusterResourceOverride":{"configuration":{"apiVersion":"v1","kind":"ClusterResourceOverrideConfig","memoryRequestToLimitPercent":"25","cpuRequestToLimitPercent":"25","limitCPUToMemoryPercent":"200"}}}` In this value, **openshift_master_admission_plugin_config={"openshift.io/ImagePolicy":{"configuration":{"apiVersion":"v1","executionRules":[{"matchImageAnnotations":[{"key":"images.openshift.io/deny-execution","value":"true"}],"name":"execution-denied","onResources":[{"resource":"pods"},{"resource":"builds"}],"reject":true,"skipOnResolutionFailure":true}],"kind":"ImagePolicyConfig"}}}** is the default parameter value. IMPORTANT You must include the default **openshift_master_admission_plugin_config** value even if you need to add a custom setting.
openshift_master_audit_config	This variable enables API service auditing. See Audit Configuration for more information.

Variable	Purpose
openshift_master_cluster_hostname	This variable overrides the host name for the cluster, which defaults to the host name of the master.
openshift_master_cluster_public_hostname	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. For example: `openshift_master_cluster_public_hostname=openshift-ansible.public.example.com`
openshift_master_cluster_method	Optional. This variable defines the HA method when deploying multiple masters. Supports the **native** method. See Multiple Masters for more information.
openshift_rolling_restart_mode	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 restarts of the master nodes. A rolling restart of the masters can be necessary to apply additional changes using the supplied Ansible hooks during the upgrade. Depending on the tasks you choose to perform you might want to reboot the host to restart your services.
openshift_master_identity_providers	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. You can configure multiple identity providers.
openshift_master_named_certificates	These variables are used to configure custom certificates which are deployed as part of the installation. See Configuring Custom Certificates for more information.
openshift_master_overwrite_named_certificates	
openshift_hosted_router_certificate	Provide the location of the custom certificates for the hosted router.
openshift_master_ca_certificate	Provide the single certificate and key that signs the OpenShift Container Platform certificates. See Redeploying a New or Custom OpenShift Container Platform CA

Variable	Purpose
openshift_additional_ca	If the certificate for your **openshift_master_ca_certificate** 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
openshift_hosted_registry_cert_expire_days	Validity of the auto-generated registry certificate in days. Defaults to **730** (2 years).
openshift_ca_cert_expire_days	Validity of the auto-generated CA certificate in days. Defaults to **1825** (5 years).
openshift_node_cert_expire_days	Validity of the auto-generated node certificate in days. Defaults to **730** (2 years).
openshift_master_cert_expire_days	Validity of the auto-generated master certificate in days. Defaults to **730** (2 years).
etcd_ca_default_days	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).
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.

Variable	Purpose
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.
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. Starting in OpenShift Container Platform 3.9, this defaults to **node-role.kubernetes.io/compute=true** if undefined.
openshift_docker_additional_registries	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>**. For example: `openshift_docker_additional_registries=example.com:443` NOTE If you need to configure your cluster to use an alternate registry, set **oreg_url** rather than rely on **openshift_docker_additional_registries**.

Variable	Purpose
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.
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 Service (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. The value for this variable must consist of lower case alphanumeric characters or dashes (**-**). It must start with an alphabetic character, and end with an alphanumeric character.
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.
osm_cluster_network_cidr	This variable overrides the SDN cluster network CIDR block. This is the network from which pod IPs are assigned. This network block should be a private block and must not conflict with existing network blocks in your infrastructure to which pods, nodes, or the master may 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.
openshift_portal_net	This variable configures the subnet in which services will be created within the OpenShift Container Platform SDN. This network block should be private and must not conflict with any existing network blocks in your infrastructure to which pods, nodes, or the master may 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.
osm_host_subnet_length	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.
openshift_node_proxy_mode	This variable specifies the service proxy mode to use: either **iptables** for the default, pure-**iptables** implementation, or **userspace** for the user space proxy.

Variable	Purpose
openshift_use_flannel	This variable enables **flannel** as an alternative networking layer instead of the default SDN. If enabling **flannel**, disable the default SDN with the **openshift_use_openshift_sdn** variable. For more information, see Using Flannel.
openshift_use_openshift_sdn	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, indicate the desired variables 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
openshift_public_hostname	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).
openshift_public_ip	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).
openshift_node_labels	This variable is deprecated; see Defining Node Groups and Host Mappings for the current method of setting node labels.

Variable	Purpose
openshift_docker_options	This variable configures additional **docker** options within */etc/sysconfig/docker*, such as options used in Managing Container Logs. It is recommended to use **json-file**. The following example shows the configuration of Docker to use the **json-file** log driver, where Docker rotates between three 1 MB log files and signature verification is not required. When supplying additional options, ensure that you maintain the single quotation mark formatting: `OPTIONS=' --selinux-enabled --log-opt  max-size=1M --log-opt max-file=3 --insecure-registry 172.30.0.0/16 --log-driver=json-file --signature-verification=false'`
openshift_schedulable	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.
openshift_node_problem_detector_install	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

Starting in OpenShift Container Platform 3.10, node configurations are now 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**

The following ConfigMaps are **CRI-O** variants for each of the existing default node groups:

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

IMPORTANT

Changes should not be made to a node host's ***/etc/origin/node/node-config.yaml*** file. They will be overwritten by the configuration defined in the ConfigMap used by the node.

4.5.2. Node Group Definitions

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

```
openshift_node_groups:
  - name: node-config-master 1
    labels:
      - 'node-role.kubernetes.io/master=true' 2
    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, the defaults defined above will be used. However, if you want to deviate from these defaults, you must define the entire **openshift_node_groups** structure (including all desired node groups) in your inventory file.

The **openshift_node_groups** value is not merged with the defaults, and the YAML definition must first be translated into a Python dictionary. You can then use the **edits** field to modify any node configuration variables as desired by specifying the 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 use 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 are using 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 exmaple:

```
[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

Labels can be assigned to node hosts during cluster installation, which are useful for determining the placement of pods onto nodes using the scheduler. While previously node labels could be set using the **openshift_node_labels** variable, starting in OpenShift Container Platform 3.10 you must instead create your own custom node groups if you want to modify the default labels that are assigned to node hosts. See Node Group Definitions for details on modifying 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

Any hosts you designate as masters during the installation process should also be configured as nodes so that the masters are configured as part of the OpenShift SDN. You must do so by adding entries for these 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 would fail to deploy because no nodes with the **node-role.kubernetes.io/compute=true** label would be 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 are using an image registry other than the default **registry.access.redhat.com**, you must specify the desired 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 PROJECT PARAMETERS

To configure the default project settings, configure the following variables in the ***/etc/ansible/hosts*** file:

Table 4.4. Project Parameters

Parameter	Description	Type	Default Value
osm_project_request_message	The string presented to a user if they are unable to request a project via the **projectrequest** API endpoint.	String	null
osm_project_request_template	The template to use for creating projects in response to a **projectrequest**. If you do not specify a value, the default template is used.	String with the format **<namespace>/<template>**	null

Parameter	Description	Type	Default Value
osm_mcs_allocator_range	Defines the range of MCS categories to assign to namespaces. If this value is changed after startup, new projects might receive labels that are already allocated to other projects. The prefix can be any valid SELinux set of terms, including user, role, and type. However, leaving the prefix at its default allows the server to set them automatically. For example, **s0:/2** allocates labels from s0:c0,c0 to s0:c511,c511 whereas **s0:/2,512** allocates labels from s0:c0,c0,c0 to s0:c511,c511,511.	String with the format **<prefix>/<numberOfLabels>[,<maxCategory>]**	**s0:/2**
osm_mcs_labels_per_project	Defines the number of labels to reserve per project.	Integer	**5**
osm_uid_allocator_range	Defines the total set of Unix user IDs (UIDs) automatically allocated to projects and the size of the block that each namespace gets. For example, **1000-1999/10** allocates ten UIDs per namespace and can allocate up to 100 blocks before running out of space. The default value is the expected size of the ranges for container images when user namespaces are started.	String in the format **<block_range>/<number_of_UIDs>**	**1000000000-1999999999/10000**

4.7. 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 4.5. 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**).

4.8. 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 4.6. 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 may be set by setting the **openshift_check_min_host_memory_gb** cluster variable in your inventory file.
disk_availability	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 may be set by setting **openshift_check_min_host_disk_gb** cluster variable in your inventory file.

Check Name	Purpose
docker_storage	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 may be set by setting the **max_thinpool_data_usage_percent** cluster variable in your inventory file.
docker_storage_driver	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.
docker_image_availability	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.
package_version	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.
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.

4.9. CONFIGURING A REGISTRY LOCATION

If you are using an image registry other than the default at **registry.access.redhat.com**, specify the desired registry within the ***/etc/ansible/hosts*** file.

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

Table 4.7. Registry Variables

Variable	Purpose
oreg_url	Set to the alternate image location. Necessary if you are not using the default registry at **registry.access.redhat.com**. The default component inherits the image prefix and version from the **oreg_url** value.
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**.

For example:

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

4.10. CONFIGURING A REGISTRY ROUTE

To allow users to push and pull images to the internal Docker 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 4.8. Registry Route Variables

Variable	Purpose

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.
openshift_hosted_registry_routecertificates	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: ● **certfile** ● **keyfile** ● **cafile**
openshift_hosted_registry_routetermination	Set to one of the following values: ● 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'}"
```

4.11. CONFIGURING THE REGISTRY CONSOLE

If you are using a Cockpit registry console image other than the default or require a specific version of the console, specify the desired registry within the ***/etc/ansible/hosts*** file:

```
openshift_cockpit_deployer_prefix=<registry_name>/<namespace>/
openshift_cockpit_deployer_version=<cockpit_image_tag>
```

Table 4.9. Registry Variables

Variable	Purpose
openshift_cockpit_deployer_prefix	Specify the URL and path to the directory where the image is located. The value for the path must end in **/openshift3** rather than **ose-**, which is the standard for other images.
openshift_cockpit_deployer_version	Specify the Cockpit image version.

For example: If your image is at **registry.example.com/openshift3/registry-console** and you require version 3.10.1, enter:

```
openshift_cockpit_deployer_prefix='registry.example.com/openshift3/'
openshift_cockpit_deployer_version='3.10.1'
```

4.12. 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']}]
```

4.13. 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.

4.13.1. 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"
```

A valid image tag is required for your deployment to succeed. Replace **<tag>** with the version of Red Hat Gluster Storage that is compatible with OpenShift Container Platform 3.10 as described in the interoperability matrix for the following variables in your inventory file:

- **openshift_storage_glusterfs_image=registry.redhat.io/rhgs3/rhgs-server-rhel7:<tag>**
- **openshift_storage_glusterfs_block_image=registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7:<tag>**

- **openshift_storage_glusterfs_s3_image=registry.redhat.io/rhgs3/rhgs-s3-server-rhel7:<tag>**
- **openshift_storage_glusterfs_heketi_image=registry.redhat.io/rhgs3/rhgs-volmanager-rhel7:<tag>**
- **openshift_storage_glusterfs_registry_image=registry.redhat.io/rhgs3/rhgs-server-rhel7:<tag>**
- **openshift_storage_glusterfs_block_registry_image=registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7:<tag>**
- **openshift_storage_glusterfs_s3_registry_image=registry.redhat.io/rhgs3/rhgs-s3-server-rhel7:<tag>**
- **openshift_storage_glusterfs_heketi_registry_image=registry.redhat.io/rhgs3/rhgs-volmanager-rhel7:<tag>**

4.13.2. 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" ]'
```

4.14. CONFIGURING AN OPENSHIFT CONTAINER REGISTRY

An integrated OpenShift Container Registry can be deployed using the installer.

4.14.1. 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 RHEL NFS server as a storage backend for the container image registry. This includes the OpenShift Container Registry and Quay. Therefore, using the RHEL NFS server 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 within the **[nfs]** host group. For example, the volume path using these options would be ***/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 would be ***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 are using 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
```

4.15. 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 4.10. 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 specifices the **HTTPS_PROXY** environment variable for masters and the Docker daemon.

Variable	Purpose
openshift_no_proxy	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. The host names that do not use the defined proxy include: ● 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	This boolean variable specifies whether or not the names of all defined OpenShift hosts and ***.cluster.local** should be automatically appended to the **NO_PROXY** list. Defaults to **true**; set it to **false** to override this option.
openshift_builddefaults_http_proxy	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.

Variable	Purpose
openshift_builddefaults_https_proxy	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 **openshift_http_proxy** parameter, the **openshift_https_proxy** value is used. Set the **openshift_builddefaults_https_proxy** value to **False** to disable default https proxy for builds regardless of the **openshift_https_proxy** value.
openshift_builddefaults_no_proxy	This variable defines the **NO_PROXY** environment variable inserted into builds using the **BuildDefaults** admission controller. Set the **openshift_builddefaults_no_proxy** value to **False** to disable default no proxy settings for builds regardless of the **openshift_no_proxy** value.
openshift_builddefaults_git_http_proxy	This variable defines the HTTP proxy used by **git clone** operations during a build, defined using the **BuildDefaults** admission controller. Set the **openshift_builddefaults_git_http_proxy** value to **False** to disable default http proxy for**git clone** operations during a build regardless of the **openshift_http_proxy** value.
openshift_builddefaults_git_https_proxy	This variable defines the HTTPS proxy used by **git clone** operations during a build, defined using the **BuildDefaults** admission controller. Set the **openshift_builddefaults_git_https_proxy** value to **False** to disable default https proxy for**git clone** operations during a build regardless of the **openshift_https_proxy** value.

4.16. 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 1
```

1 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.

4.17. 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+pmZNSO']
```

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+pmZNSO']
```

4.18. 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

Custom certificates should only be configured for the host name associated with the **publicMasterURL** which can be set using **openshift_master_cluster_public_hostname**. Using a custom serving certificate for the host name associated with the **masterURL** (**openshift_master_cluster_hostname**) will result in TLS errors as infrastructure components will 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 within 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 within the master configuration file.

If you would like **openshift_master_named_certificates** to be overwritten 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, you could set the following:

```
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
```

4.19. 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.

4.20. 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.

4.20.1. 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 RHEL NFS server as a storage backend for the container image registry. This includes Cassandra for metrics storage. Therefore, using the RHEL NFS server 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. Use the following variables:

```
[OSEv3:vars]

openshift_metrics_cassandra_storage_type=pv
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 within the **[nfs]** host group. For example, the volume path using these options would be ***/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 would be ***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
```

4.21. 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
```

4.21.1. 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 RHEL NFS server as a storage backend for the container image registry. This includes ElasticSearch for logging storage. Therefore, using the RHEL NFS server 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 has dynamic volume provisioning, it could be configured either via the cloud provider or by an independent storage provider. For instance, the cloud provider could have a StorageClass with provisioner **kubernetes.io/gce-pd** on GCE, and an independent storage provider such as GlusterFS could have a **StorageClass** with provisioner **kubernetes.io/glusterfs**. In either case, use the following variable:

```
[OSEv3:vars]

openshift_logging_es_pvc_dynamic=true
```

For additional information on dynamic provisioning, see Dynamic provisioning and creating storage classes.

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. Use the following variables:

```
[OSEv3:vars]

openshift_logging_elasticsearch_storage_type=pvc
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 within the **[nfs]** host group. For example, the volume path using these options would be ***/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 would be ***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
```

4.22. 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"}
```

4.22.1. 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 4.11. Service broker customization variables

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

4.22.1.1. 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 within the **[nfs]** group. For example, the volume path using these options would be */opt/osev3-etcd/etcd-vol2*.

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

4.22.1.2. 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$']
```

4.22.2. 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 desired projects by modifying the following in your inventory file's **[OSEv3:vars]** section:

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

By default, the TSB will use the nodeselector **{"node-role.kubernetes.io/infra":"true"}** for deploying its pods. You can modify this by setting the desired nodeselector in your inventory file's **[OSEv3:vars]** section:

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

Table 4.12. 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.

4.23. 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 4.13. 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
openshift_master_logout_url	Sets **clusterInfo.logoutPublicURL** in the web console configuration. See Changing the Logout URL for details. Example value: **https://example.com/logout**
openshift_web_console_extension_script_urls	Sets **extensions.scriptURLs** in the web console configuration. See Loading Extension Scripts and Stylesheets for details. Example value: **['https://example.com/scripts/menu-customization.js','https://example.com/scripts/nav-customization.js']**
openshift_web_console_extension_stylesheet_urls	Sets **extensions.stylesheetURLs** in the web console configuration. See Loading Extension Scripts and Stylesheets for details. Example value: **['https://example.com/styles/logo.css','https://example.com/styles/custom-styles.css']**
openshift_master_oauth_template	Sets the OAuth template in the master configuration. See Customizing the Login Page for details. Example value: **['/path/to/login-template.html']**
openshift_master_metrics_public_url	Sets **metricsPublicURL** in the master configuration. See Setting the Metrics Public URL for details. Example value: **https://hawkular-metrics.example.com/hawkular/metrics**
openshift_master_logging_public_url	Sets **loggingPublicURL** in the master configuration. See Kibana for details. Example value: **https://kibana.example.com**
openshift_web_console_inactivity_timeout_minutes	Configurate 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).
openshift_web_console_cluster_resource_overrides_enabled	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 since these values should be set by the cluster resource override configuration.

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, the **openshift_node_groups** variable must also be set; 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 external 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

# 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

# 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.

- 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

# 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 availbility 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']}]}]

# 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'
```

```
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

# 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. RUNNING INSTALLATION PLAYBOOKS

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

IMPORTANT

Running Ansible playbooks with the **--tags** or **--check** options is not supported by Red Hat.

6.1. BEFORE INITIATING INSTALLATION

Before installing OpenShift Container Platform, you must first:

- See the Prerequisites and Host Preparation topics to prepare your hosts. This includes verifying system and environment requirements per component type and properly installing and configuring the **docker** service. It also includes installing Ansible version 2.4 or later, as the installation method is based on Ansible playbooks and as such requires directly invoking Ansible.
- See the Configuring Your Inventory File topic to define your environment and desired OpenShift Container Platform cluster configuration. This inventory file will be used to initiate the installation, and should be saved and maintained for future cluster upgrades as well.

IMPORTANT

Starting in OpenShift Container Platform 3.10, setting **openshift_node_group_name** per host to a node group is required for all cluster installations whether you are using the default node group definitions and ConfigMaps or are customizing your own. See Defining Node Groups and Host Mappings for more details if you have not set them yet.

If you are interested in installing OpenShift Container Platform using the system container method (required for RHEL Atomic Host systems), see RPM Versus System Container Considerations to ensure that you understand the differences between these methods, then return to this topic to continue.

For large-scale installs, including suggestions for optimizing install time, see the Scaling and Performance Guide.

NOTE

To alternatively install OpenShift Container Platform solely as a stand-alone registry, see Installing a Stand-alone Registry .

6.2. RUNNING THE INSTALLATION PLAYBOOKS

The installer uses modularized playbooks allowing administrators to 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 playbooks and their ordering are detailed below in Running Individual Component Playbooks.

IMPORTANT

While RHEL Atomic Host is supported for running OpenShift Container Platform services as system container, the installation method utilizes 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.

After you have configured Ansible by defining an inventory file in ***/etc/ansible/hosts***, run the installation playbook via Ansible using either the RPM-based or containerized installer.

NOTE

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.

6.2.1. 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 and not closed. Therefore, the system can run out of files to open and the playbook will fail.

To run the RPM-based installer:

1. 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:

```
# ansible-playbook [-i /path/to/inventory] \ 1
    /usr/share/ansible/openshift-ansible/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. Run the ***deploy_cluster.yml*** playbook to initiate the cluster installation:

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

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

WARNING

The installer caches playbook configuration values for 10 minutes, by default. If you change any system, network, or inventory configuration, and then re-run the installer within that 10 minute period, the new values are not used, and the previous values are used instead. You can delete the contents of the cache, which is defined by the **fact_caching_connection** value in the ***/etc/ansible/ansible.cfg*** file. An example of this file is shown in Recommended Installation Practices.

6.2.2. 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.

6.2.2.1. 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 \ 1
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml \
    --set OPTS="-v" \
    registry.access.redhat.com/openshift3/ose-ansible:v3.10
```

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

This command runs a set of prerequiste 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 \ 1
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml
\
    --set OPTS="-v" \
    registry.access.redhat.com/openshift3/ose-ansible:v3.10
```

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.

6.2.2.2. 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.access.redhat.com/openshift3/ose-ansible:v3.10
```

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**.

6.2.2.3. 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, as the install may restart **docker** on the host, disrupting the installer container execution.

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` \ ①
    -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \ ②
    -v $HOME/ansible/hosts:/tmp/inventory:Z \ ③
    -e INVENTORY_FILE=/tmp/inventory \ ④
    -e PLAYBOOK_FILE=playbooks/prerequisites.yml \ ⑤
    -e OPTS="-v" \ ⑥
    registry.access.redhat.com/openshift3/ose-ansible:v3.10
```

① **-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).

② **-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 **$HOME/.ssh** (*/opt/app-root/src* is the **$HOME** of the user in the container). If you mount the SSH key into a non-standard location you can 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 at it. Note that the SSH key is mounted with the **:Z** flag. This 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**, check 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 login. For this reason you may want to use a separate copy of the SSH key (or directory), so that the original file labels remain untouched.

3 4 **-v $HOME/ansible/hosts:/tmp/inventory:Z** and **-e INVENTORY_FILE=/tmp/inventory** mount a static Ansible inventory file into the container as ***/tmp/inventory*** and set the corresponding environment variable to point at it. As with the SSH key, the inventory file SELinux labels may need to be relabeled by using the **:Z** flag to allow reading in the container, depending on the existing label (for files in a user **$HOME** directory this is likely to be needed). So again you may prefer to copy the inventory to a dedicated location before mounting it. The inventory file can also be downloaded from a web server if you specify the **INVENTORY_URL** environment variable, or generated dynamically using **DYNAMIC_SCRIPT_URL** to specify an executable script that provides a dynamic inventory.

5 **-e PLAYBOOK_FILE=playbooks/prerequisites.yml** specifies the playbook to run (in this example, the prereqsuites playbook) as a relative path from the top level directory of **openshift-ansible** content. The full path from the RPM can also be used, as well as the path to any other playbook file in the container.

6 **-e OPTS="-v"** supplies arbitrary command line options (in this case, **-v** to increase verbosity) to the **ansible-playbook** command that runs inside the container.

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.access.redhat.com/openshift3/ose-ansible:v3.10
```

6.2.2.4. Running the Installation Playbook for OpenStack

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
```

6.2.3. Running Individual Component Playbooks

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.

After you resolve the errors, you can continue installation:

- You can run the remaining individual installation playbooks.
- If you are installing in a new environment, you can run the ***deploy_cluster.yml*** playbook again.

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:

```
# 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	***/usr/share/ansible/openshift-ansible/playbooks/openshift-checks/pre-install.yml***
Node Bootstrap	***/usr/share/ansible/openshift-ansible/playbooks/openshift-node/bootstrap.yml***
etcd Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-etcd/config.yml***
NFS Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-nfs/config.yml***
Load Balancer Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-loadbalancer/config.yml***
Master Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-master/config.yml***
Master Additional Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-master/additional_config.yml***
Node Join	***/usr/share/ansible/openshift-ansible/playbooks/openshift-node/join.yml***
GlusterFS Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-glusterfs/config.yml***
Hosted Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-hosted/config.yml***
Monitoring Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-monitoring/config.yml***
Web Console Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-web-console/config.yml***
Metrics Install	***/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml***

Playbook Name	File Location
Logging Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml*
Prometheus Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-prometheus/config.yml*
Availability Monitoring Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-monitor-availability/config.yml*
Service Catalog Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-service-catalog/config.yml*
Management Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-management/config.yml*
Descheduler Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-descheduler/config.yml*
Node Problem Detector Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-node-problem-detector/config.yml*
Autoheal Install	*/usr/share/ansible/openshift-ansible/playbooks/openshift-autoheal/config.yml*

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 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 would be found at **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:237
9 \
    --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:237
9 \
    --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, browse to the following URL according to your **[lb]** section definition and check HAProxy's status:

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

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 use a more secure option at the time of 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 on this optional process.

6.5. UNINSTALLING OPENSHIFT 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 will delete content for any hosts defined in the inventory file that you specify when running the playbook.

IMPORTANT

Before you uninstall your cluster, review the following list of scenarios and make sure that uninstalling is the best option:

- If your installation process failed and you want to continue the process, you can retry the installation. The installation playbooks are designed so that if they fail to install your cluster, you can run them again without needing to uninstall the cluster.

- If you want to restart a failed installation from the beginning, you can uninstall the OpenShift Container Platform hosts in your cluster by running the ***uninstall.yml*** playbook, as described in the following section. This playbook only uninstalls the OpenShift Container Platform assets for the most recent version that you installed.

- If you must change the host names or certificate names, you must recreate your certificates before retrying installation by running the ***uninstall.yml*** playbook. Running the installation playbooks again will not recreate the certificates.

- If you want to repurpose hosts that you installed OpenShift Container Platform on earlier, such as with a proof-of-concept installation, or want to install a different minor or asynchronous version of OpenShift Container Platform you must reimage the hosts before you use them in a production cluster. After you run the ***uninstall.yml*** playbooks, some host assets might remain in an altered state.

If you want to uninstall OpenShift Container Platform across all hosts in your cluster, run the playbook using the inventory file you used when installing OpenShift Container Platform initially or ran most recently:

```
# ansible-playbook [-i /path/to/file] \
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

6.5.1. Uninstalling Nodes

You can also uninstall node components from specific hosts using the ***uninstall.yml*** playbook while leaving the remaining hosts and cluster alone:

WARNING

This method should only be used when attempting to uninstall specific node hosts and not for specific masters or etcd hosts, which would require further configuration changes within the cluster.

1. First follow the steps in Deleting Nodes to remove the node object from the cluster, then continue with the remaining steps in this procedure.

2. Create a different inventory file that only references those hosts. For example, to only delete content from 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) Only include the sections that pertain to the hosts you are interested in uninstalling.

(2) Only include hosts that you want to uninstall.

3. Specify that new inventory file using the **-i** option when running the ***uninstall.yml*** playbook:

```
# ansible-playbook -i /path/to/new/file \
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

When the playbook completes, all OpenShift Container Platform content should be removed from any specified hosts.

6.6. 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.

- If the Ansible installer fails, you can still install OpenShift Container Platform:
  - If you did not modify the SDN configuration or generate new certificates, run the ***deploy_cluster.yml*** playbook again.
  - 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 fresh image or uninstall and install again.
  - If you use bare metal machines, uninstall and install again.

6.7. WHAT'S NEXT?

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

- Deploy an integrated Docker 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 Docker 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.

IMPORTANT

You must obtain the required images and software components on a system with the same architecture as the cluster that is in your disconnected environment.

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="*"
      ```

5. Enable only the repositories required by OpenShift Container Platform 3.10.

   - For cloud installations and on-premise installations on x86_64 servers, run the following command:

     ```
     # subscription-manager repos \
         --enable="rhel-7-server-rpms" \
         --enable="rhel-7-server-extras-rpms" \
         --enable="rhel-7-server-ose-3.10-rpms" \
         --enable="rhel-7-server-ansible-2.4-rpms"
     ```

   - For on-premise installations on IBM POWER8 servers, run the following command:

     ```
     # subscription-manager repos \
         --enable="rhel-7-for-power-le-rpms" \
         --enable="rhel-7-for-power-le-extras-rpms" \
         --enable="rhel-7-for-power-le-optional-rpms" \
         --enable="rhel-7-server-ansible-2.6-for-power-le-rpms" \
         --enable="rhel-7-server-for-power-le-rhscl-rpms" \
         --enable="rhel-7-for-power-le-ose-3.10-rpms"
     ```

   - For on-premise installations on IBM POWER9 servers, run the following command:

     ```
     # subscription-manager repos \
         --enable="rhel-7-for-power-9-rpms" \
         --enable="rhel-7-for-power-9-extras-rpms" \
         --enable="rhel-7-for-power-9-optional-rpms" \
         --enable="rhel-7-server-ansible-2.6-for-power-9-rpms" \
         --enable="rhel-7-server-for-power-le-rhscl-rpms" \
         --enable="rhel-7-for-power-le-ose-3.10-rpms"
     ```

6. 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.

7. 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.

8. Sync the packages and create the repository for each of them.

- For on-premise installations on x86_64 servers, run the following command:

```
$ for repo in \
rhel-7-server-rpms \
rhel-7-server-extras-rpms \
rhel-7-server-ansible-2.4-rpms \
rhel-7-server-ose-3.10-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.

- For on-premise installations on IBM POWER8 servers, run the following command:

```
$ for repo in \
rhel-7-for-power-le-rpms \
rhel-7-for-power-le-extras-rpms \
rhel-7-for-power-le-optional-rpms \
rhel-7-server-ansible-2.6-for-power-le-rpms \
rhel-7-server-for-power-le-rhscl-rpms \
rhel-7-for-power-le-ose-3.10-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.

- For on-premise installations on IBM POWER9 servers, run the following command:

```
$ for repo in \
rhel-7-for-power-9-rpms \
rhel-7-for-power-9-extras-rpms \
rhel-7-for-power-9-optional-rpms \
rhel-7-server-ansible-2.6-for-power-9-rpms \
rhel-7-server-for-power-le-rhscl-rpms \
rhel-7-for-power-le-ose-3.10-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=/<path/to/repos> ❶
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} ❷
done
```

❶ ❷ 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 **v3.10.181** for the latest version. You can specify a different minor version.

```
$ docker pull registry.access.redhat.com/openshift3/csi-attacher:<tag>
$ docker pull registry.access.redhat.com/openshift3/csi-driver-registrar:<tag>
$ docker pull registry.access.redhat.com/openshift3/csi-livenessprobe:<tag>
$ docker pull registry.access.redhat.com/openshift3/csi-provisioner:<tag>
$ docker pull registry.access.redhat.com/openshift3/image-inspector:<tag>
$ docker pull registry.access.redhat.com/openshift3/local-storage-provisioner:<tag>
$ docker pull registry.access.redhat.com/openshift3/manila-provisioner:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-ansible:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-cli:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-cluster-capacity:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-deployer:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-descheduler:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-docker-builder:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-docker-registry:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-egress-dns-proxy:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-egress-http-proxy:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-egress-router:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-f5-router:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-haproxy-router:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-hyperkube:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-hypershift:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-keepalived-ipfailover:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-pod:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-node-problem-detector:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-recycler:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-web-console:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-node:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-control-plane:<tag>
$ docker pull registry.access.redhat.com/openshift3/registry-console:<tag>
$ docker pull registry.access.redhat.com/openshift3/snapshot-controller:<tag>
$ docker pull registry.access.redhat.com/openshift3/snapshot-provisioner:<tag>
$ docker pull registry.access.redhat.com/openshift3/apb-base:<tag>
$ docker pull registry.access.redhat.com/openshift3/apb-tools:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-service-catalog:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-ansible-service-broker:<tag>
$ docker pull registry.access.redhat.com/openshift3/mariadb-apb:<tag>
$ docker pull registry.access.redhat.com/openshift3/mediawiki-apb:<tag>
$ docker pull registry.access.redhat.com/openshift3/mysql-apb:<tag>
$ docker pull registry.access.redhat.com/openshift3/ose-template-service-broker:<tag>
$ docker pull registry.access.redhat.com/openshift3/postgresql-apb:<tag>
$ docker pull registry.access.redhat.com/rhel7/etcd:3.2.22
```

3. For on-premise installations on x86_64 servers, pull the following image. Replace **<tag>** with the version to install. For example, specify **v3.10.181** for the latest version. You can specify a different minor version.

```
$ docker pull registry.access.redhat.com/openshift3/efs-provisioner:<tag>
```

4. Pull all of the required OpenShift Container Platform component images for the optional components. Replace **<tag>** with the version to install. For example, specify **v3.10.181** for the latest version. You can specify a different minor version.

   - For on-premise installations on x86_64 servers, run the following commands:

```
$ docker pull registry.access.redhat.com/openshift3/logging-auth-proxy:<tag>
$ docker pull registry.access.redhat.com/openshift3/logging-curator:<tag>
$ docker pull registry.access.redhat.com/openshift3/logging-elasticsearch:<tag>
$ docker pull registry.access.redhat.com/openshift3/logging-eventrouter:<tag>
$ docker pull registry.access.redhat.com/openshift3/logging-fluentd:<tag>
$ docker pull registry.access.redhat.com/openshift3/logging-kibana:<tag>
$ docker pull registry.access.redhat.com/openshift3/oauth-proxy:<tag>
$ docker pull registry.access.redhat.com/openshift3/metrics-cassandra:<tag>
$ docker pull registry.access.redhat.com/openshift3/metrics-hawkular-metrics:<tag>
$ docker pull registry.access.redhat.com/openshift3/metrics-hawkular-openshift-agent:
<tag>
$ docker pull registry.access.redhat.com/openshift3/metrics-heapster:<tag>
$ docker pull registry.access.redhat.com/openshift3/metrics-schema-installer:<tag>
$ docker pull registry.access.redhat.com/openshift3/prometheus:<tag>
$ docker pull registry.access.redhat.com/openshift3/prometheus-alert-buffer:<tag>
$ docker pull registry.access.redhat.com/openshift3/prometheus-alertmanager:<tag>
$ docker pull registry.access.redhat.com/openshift3/prometheus-node-exporter:<tag>
$ docker pull registry.access.redhat.com/cloudforms46/cfme-openshift-postgresql
$ docker pull registry.access.redhat.com/cloudforms46/cfme-openshift-memcached
$ docker pull registry.access.redhat.com/cloudforms46/cfme-openshift-app-ui
$ docker pull registry.access.redhat.com/cloudforms46/cfme-openshift-app
$ docker pull registry.access.redhat.com/cloudforms46/cfme-openshift-embedded-
ansible
$ docker pull registry.access.redhat.com/cloudforms46/cfme-openshift-httpd
$ docker pull registry.access.redhat.com/cloudforms46/cfme-httpd-configmap-generator
$ docker pull registry.access.redhat.com/rhgs3/rhgs-server-rhel7
$ docker pull registry.access.redhat.com/rhgs3/rhgs-volmanager-rhel7
$ docker pull registry.access.redhat.com/rhgs3/rhgs-gluster-block-prov-rhel7
$ docker pull registry.access.redhat.com/rhgs3/rhgs-s3-server-rhel7
```

- For on-premise installations on IBM POWER8 or IBM POWER9 servers, run the following commands:

  ```
  $ docker pull registry.access.redhat.com/openshift3/logging-auth-proxy:<tag>
  $ docker pull registry.access.redhat.com/openshift3/logging-curator:<tag>
  $ docker pull registry.access.redhat.com/openshift3/logging-elasticsearch:<tag>
  $ docker pull registry.access.redhat.com/openshift3/logging-eventrouter:<tag>
  $ docker pull registry.access.redhat.com/openshift3/logging-fluentd:<tag>
  $ docker pull registry.access.redhat.com/openshift3/logging-kibana:<tag>
  $ docker pull registry.access.redhat.com/openshift3/oauth-proxy:<tag>
  $ docker pull registry.access.redhat.com/openshift3/metrics-cassandra:<tag>
  $ docker pull registry.access.redhat.com/openshift3/metrics-hawkular-metrics:<tag>
  $ docker pull registry.access.redhat.com/openshift3/metrics-hawkular-openshift-agent:<tag>
  $ docker pull registry.access.redhat.com/openshift3/metrics-heapster:<tag>
  $ docker pull registry.access.redhat.com/openshift3/metrics-schema-installer:<tag>
  $ docker pull registry.access.redhat.com/openshift3/prometheus:<tag>
  $ docker pull registry.access.redhat.com/openshift3/prometheus-alert-buffer:<tag>
  $ docker pull registry.access.redhat.com/openshift3/prometheus-alertmanager:<tag>
  $ docker pull registry.access.redhat.com/openshift3/prometheus-node-exporter:<tag>
  ```

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/.

5. 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.access.redhat.com/jboss-amq-6/amq63-openshift
   $ docker pull registry.access.redhat.com/jboss-datagrid-7/datagrid71-openshift
   $ docker pull registry.access.redhat.com/jboss-datagrid-7/datagrid71-client-openshift
   $ docker pull registry.access.redhat.com/jboss-datavirt-6/datavirt63-openshift
   $ docker pull registry.access.redhat.com/jboss-datavirt-6/datavirt63-driver-openshift
   $ docker pull registry.access.redhat.com/jboss-decisionserver-6/decisionserver64-openshift
   $ docker pull registry.access.redhat.com/jboss-processserver-6/processserver64-openshift
   $ docker pull registry.access.redhat.com/jboss-eap-6/eap64-openshift
   ```

```
$ docker pull registry.access.redhat.com/jboss-eap-7/eap70-openshift
$ docker pull registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat7-openshift
$ docker pull registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat8-openshift
$ docker pull registry.access.redhat.com/openshift3/jenkins-1-rhel7:<tag>
$ docker pull registry.access.redhat.com/openshift3/jenkins-2-rhel7:<tag>
$ docker pull registry.access.redhat.com/openshift3/jenkins-agent-maven-35-rhel7:<tag>
$ docker pull registry.access.redhat.com/openshift3/jenkins-agent-nodejs-8-rhel7:<tag>
$ docker pull registry.access.redhat.com/openshift3/jenkins-slave-base-rhel7:<tag>
$ docker pull registry.access.redhat.com/openshift3/jenkins-slave-maven-rhel7:<tag>
$ docker pull registry.access.redhat.com/openshift3/jenkins-slave-nodejs-rhel7:<tag>
$ docker pull registry.access.redhat.com/rhscl/mongodb-32-rhel7
$ docker pull registry.access.redhat.com/rhscl/mysql-57-rhel7
$ docker pull registry.access.redhat.com/rhscl/perl-524-rhel7
$ docker pull registry.access.redhat.com/rhscl/php-56-rhel7
$ docker pull registry.access.redhat.com/rhscl/postgresql-95-rhel7
$ docker pull registry.access.redhat.com/rhscl/python-35-rhel7
$ docker pull registry.access.redhat.com/redhat-sso-7/sso70-openshift
$ docker pull registry.access.redhat.com/rhscl/ruby-24-rhel7
$ docker pull registry.access.redhat.com/redhat-openjdk-18/openjdk18-openshift
$ docker pull registry.access.redhat.com/redhat-sso-7/sso71-openshift
$ docker pull registry.access.redhat.com/rhscl/nodejs-6-rhel7
$ docker pull registry.access.redhat.com/rhscl/mariadb-101-rhel7
```

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.access.redhat.com/openshift3/csi-attacher \
    registry.access.redhat.com/openshift3/csi-driver-registrar \
    registry.access.redhat.com/openshift3/csi-livenessprobe \
    registry.access.redhat.com/openshift3/csi-provisioner \
    registry.access.redhat.com/openshift3/efs-provisioner \
    registry.access.redhat.com/openshift3/image-inspector \
    registry.access.redhat.com/openshift3/local-storage-provisioner \
    registry.access.redhat.com/openshift3/manila-provisioner \
    registry.access.redhat.com/openshift3/ose-ansible \
    registry.access.redhat.com/openshift3/ose-cli \
    registry.access.redhat.com/openshift3/ose-cluster-capacity \
    registry.access.redhat.com/openshift3/ose-deployer \
    registry.access.redhat.com/openshift3/ose-descheduler \
    registry.access.redhat.com/openshift3/ose-docker-builder \
    registry.access.redhat.com/openshift3/ose-docker-registry \
    registry.access.redhat.com/openshift3/ose-egress-dns-proxy \
    registry.access.redhat.com/openshift3/ose-egress-http-proxy \
    registry.access.redhat.com/openshift3/ose-egress-router \
    registry.access.redhat.com/openshift3/ose-f5-router \
    registry.access.redhat.com/openshift3/ose-haproxy-router \
    registry.access.redhat.com/openshift3/ose-hyperkube \
    registry.access.redhat.com/openshift3/ose-hypershift \
    registry.access.redhat.com/openshift3/ose-keepalived-ipfailover \
    registry.access.redhat.com/openshift3/ose-pod \
    registry.access.redhat.com/openshift3/ose-node-problem-detector \
    registry.access.redhat.com/openshift3/ose-recycler \
    registry.access.redhat.com/openshift3/ose-web-console \
    registry.access.redhat.com/openshift3/ose-node \
    registry.access.redhat.com/openshift3/ose-control-plane \
    registry.access.redhat.com/openshift3/registry-console \
    registry.access.redhat.com/openshift3/snapshot-controller \
    registry.access.redhat.com/openshift3/snapshot-provisioner \
    registry.access.redhat.com/openshift3/apb-base \
    registry.access.redhat.com/openshift3/apb-tools \
    registry.access.redhat.com/openshift3/ose-service-catalog \
    registry.access.redhat.com/openshift3/ose-ansible-service-broker \
    registry.access.redhat.com/openshift3/mariadb-apb \
    registry.access.redhat.com/openshift3/mediawiki-apb \
    registry.access.redhat.com/openshift3/mysql-apb \
    registry.access.redhat.com/openshift3/ose-template-service-broker \
    registry.access.redhat.com/openshift3/postgresql-apb \
    registry.access.redhat.com/rhel7/etcd:3.2.22
```

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

```
$ docker save -o ose3-optional-imags.tar \
    registry.access.redhat.com/openshift3/logging-curator5 \
    registry.access.redhat.com/openshift3/logging-elasticsearch5 \
    registry.access.redhat.com/openshift3/logging-eventrouter \
    registry.access.redhat.com/openshift3/logging-fluentd \
    registry.access.redhat.com/openshift3/logging-kibana5 \
    registry.access.redhat.com/openshift3/oauth-proxy \
    registry.access.redhat.com/openshift3/metrics-cassandra \
    registry.access.redhat.com/openshift3/metrics-hawkular-metrics \
    registry.access.redhat.com/openshift3/metrics-hawkular-openshift-agent \
    registry.access.redhat.com/openshift3/metrics-heapster \
    registry.access.redhat.com/openshift3/metrics-schema-installer \
    registry.access.redhat.com/openshift3/prometheus \
    registry.access.redhat.com/openshift3/prometheus-alert-buffer \
    registry.access.redhat.com/openshift3/prometheus-alertmanager \
    registry.access.redhat.com/openshift3/prometheus-node-exporter \
    registry.access.redhat.com/cloudforms46/cfme-openshift-postgresql \
    registry.access.redhat.com/cloudforms46/cfme-openshift-memcached \
    registry.access.redhat.com/cloudforms46/cfme-openshift-app-ui \
    registry.access.redhat.com/cloudforms46/cfme-openshift-app \
    registry.access.redhat.com/cloudforms46/cfme-openshift-embedded-ansible \
    registry.access.redhat.com/cloudforms46/cfme-openshift-httpd \
    registry.access.redhat.com/cloudforms46/cfme-httpd-configmap-generator \
    registry.access.redhat.com/rhgs3/rhgs-server-rhel7 \
    registry.access.redhat.com/rhgs3/rhgs-volmanager-rhel7 \
    registry.access.redhat.com/rhgs3/rhgs-gluster-block-prov-rhel7 \
    registry.access.redhat.com/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.access.redhat.com/jboss-webserver-3/webserver31-tomcat7-openshift \
    registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat8-openshift \
    registry.access.redhat.com/openshift3/jenkins-1-rhel7 \
    registry.access.redhat.com/openshift3/jenkins-2-rhel7 \
    registry.access.redhat.com/openshift3/jenkins-agent-maven-35-rhel7 \
    registry.access.redhat.com/openshift3/jenkins-agent-nodejs-8-rhel7 \
    registry.access.redhat.com/openshift3/jenkins-slave-base-rhel7 \
    registry.access.redhat.com/openshift3/jenkins-slave-maven-rhel7 \
    registry.access.redhat.com/openshift3/jenkins-slave-nodejs-rhel7
```

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.access.redhat.com/openshift3/ose-node:<tag>
    registry.example.com/openshift3/ose-node:<tag>
    $ docker tag registry.access.redhat.com/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.access.redhat.com/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 1
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 2
enabled=1
gpgcheck=0
[rhel-7-server-ansible-2.4-rpms]
name=rhel-7-server-ansible-2.4-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ansible-2.4-rpms 3
enabled=1
gpgcheck=0
[rhel-7-server-ose-3.10-rpms]
name=rhel-7-server-ose-3.10-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ose-3.10-rpms 4
enabled=1
gpgcheck=0
```

1 2 3 4 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 OPENSHIFT 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:

```
oreg_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 OPENSHIFT CONTAINER REGISTRY

8.1. ABOUT OPENSHIFT CONTAINER REGISTRY

OpenShift Container Platform is a fully-featured enterprise solution that includes an integrated container registry called OpenShift Container Registry (OCR). Alternatively, instead of deploying OpenShift Container Platform as a full PaaS environment for developers, you can install OCR as a stand-alone container registry to run on-premise 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 registry is deployed to run on the cluster. This stand-alone deployment option is useful for administrators that want a container 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 may want to 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.

8.2. 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 to be used for Docker's storage back end; see Configuring Docker Storage for details.

IMPORTANT

OpenShift Container Platform supports servers with x86_64 or IBM POWER architecture. If you use IBM POWER servers to host cluster nodes, you can only use IBM POWER servers.

NOTE

Meeting the */var/* file system sizing requirements in RHEL Atomic Host requires making changes to the default configuration. See Managing Storage in Red Hat Enterprise Linux Atomic Host for instructions on configuring this during or after installation.

8.3. SUPPORTED SYSTEM TOPOLOGIES

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

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

8.4. HOST PREPARATION

Before installing stand-alone OCR, all of the same steps detailed in the Host Preparation topic for installing a full OpenShift Container Platform PaaS must be performed. This includes registering and subscribing the host(s) to the proper repositories, installing or updating certain packages, and setting up Docker and its storage requirements.

Follow the steps in the Host Preparation topic, then continue to Stand-alone Registry Installation Methods.

8.5. INSTALLING USING ANSIBLE

When installing stand-alone OCR, the steps are mostly the same as installing a full OpenShift Container Platform cluster using Ansible, as described in the full cluster installation process. The main difference is that you must set **deployment_subtype=registry** in the inventory file within the **[OSEv3:vars]** section for the playbooks to follow the registry installation path.

See 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 1
openshift_hosted_infra_selector="" 2

# 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'
```

1 Set **deployment_subtype=registry** to ensure installation of stand-alone OCR and not a full OpenShift Container Platform environment.

2 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 1

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.

After you have configured Ansible by defining an inventory file in ***/etc/ansible/hosts***:

1. Run the ***prerequisites.yml*** playbook to configure base packages and Docker. This must be run only once before deploying a new cluster. Use the following command, specifying **-i** if your inventory file located somewhere other than ***/etc/ansible/hosts***:

IMPORTANT

The host that you run the Ansible playbook on must have at least 75MiB of free memory per host in the inventory.

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

2. Run the ***deploy_cluster.yml*** playbook to initiate the installation:

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

NOTE

For more detailed usage information on the cluster installation process, including a comprehensive list of available Ansible variables, see the full documentation starting with Planning Your Installation.