OpenShift Container Platform 4.17

Installing on any platform

Installing OpenShift Container Platform on any platform

Last Updated: 2026-04-01

OpenShift Container Platform 4.17 Installing on any platform

Installing OpenShift Container Platform on any platform

Legal Notice

Abstract

This document describes how to install OpenShift Container Platform on any platform.

Table of Contents

CHAPTER 1. INSTALLING A CLUSTER ON ANY PLATFORM

In OpenShift Container Platform version 4.17, you can install a cluster on any infrastructure that you provision, including virtualization and cloud environments.

IMPORTANT

Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you attempt to install an OpenShift Container Platform cluster in virtualized or cloud environments.

1.1. PREREQUISITES

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

NOTE

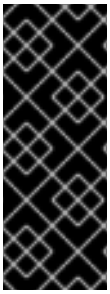
Be sure to also review this site list if you are configuring a proxy.

1.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.17, you require access to the internet to install your cluster.

You must have internet access to perform the following actions:

- Access OpenShift Cluster Manager to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access Quay.io to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.



IMPORTANT

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

1.3. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

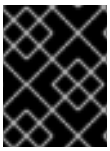
For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

This section describes the requirements for deploying OpenShift Container Platform on user-provisioned infrastructure.

1.3.1. Required machines for cluster installation

You must specify the minimum required machines or hosts for your cluster so that your cluster remains stable if a node fails.

The smallest OpenShift Container Platform clusters require the following hosts:



IMPORTANT

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

Table 1.1. Minimum required hosts

Hosts	Description
One temporary bootstrap machine	The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.
Three control plane machines	The control plane machines run the Kubernetes and OpenShift Container Platform services that form the control plane.
At least two compute machines, which are also known as worker machines.	The workloads requested by OpenShift Container Platform users run on the compute machines.

IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system. However, the compute machines can choose between Red Hat Enterprise Linux CoreOS (RHCOS), Red Hat Enterprise Linux (RHEL) 8.6 and later.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 9.2 and inherits all of its hardware certifications and requirements. See Red Hat Enterprise Linux technology capabilities and limits .

1.3.2. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

Table 1.2. Minimum resource requirements

Machine	Operating System	vCPU [1]	Virtual RAM	Storage	Input/Output Per Second (IOPS)[2]
Bootstrap	RHCOS	4	16 GB	100 GB	300
Control plane	RHCOS	4	16 GB	100 GB	300
Compute	RHCOS, RHEL 8.6 and later [3]	2	8 GB	100 GB	300

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

As of OpenShift Container Platform version 4.13, RHCOS is based on RHEL version 9.2, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

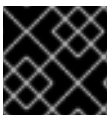
1.3.3. Certificate signing requests management

On user-provisioned infrastructure, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation when your cluster has limited access to automatic machine management.

The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

1.3.4. Networking requirements for user-provisioned infrastructure

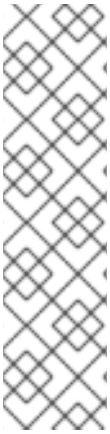
You must configure networking for all the Red Hat Enterprise Linux CoreOS (RHCOS) machines in **initramfs** during boot, so that they can fetch their Ignition config files.



IMPORTANT

Ensure you enable the **disk.EnableUUID** parameter on all virtual machines in your cluster.

During the initial boot, the machines require an IP address configuration that is set either through a DHCP server or statically by providing the required boot options. After a network connection is established, the machines download their Ignition config files from an HTTP or HTTPS server. The Ignition config files are then used to set the exact state of each machine. The Machine Config Operator completes more changes to the machines, such as the application of new certificates or keys, after installation.



NOTE

- Consider using a DHCP server for long-term management of the cluster machines. Ensure that the DHCP server is configured to provide persistent IP addresses, DNS server information, and hostnames to the cluster machines.
- If a DHCP service is not available for your user-provisioned infrastructure, you can instead provide the IP networking configuration and the address of the DNS server to the nodes at RHCOS install time. These can be passed as boot arguments if you are installing from an ISO image. See the *Installing RHCOS and starting the OpenShift Container Platform bootstrap process* section for more information about static IP provisioning and advanced networking options.

The Kubernetes API server must be able to resolve the node names of the cluster machines. If the API servers and worker nodes are in different zones, you can configure a default DNS search zone to allow the API server to resolve the node names. Another supported approach is to always refer to hosts by their fully-qualified domain names in both the node objects and all DNS requests.

1.3.4.1. Setting the cluster node hostnames through DHCP

On Red Hat Enterprise Linux CoreOS (RHCOS) machines, the hostname is set through NetworkManager. By default, the machines obtain their hostname through DHCP. If the hostname is not provided by DHCP, set statically through kernel arguments, or another method, it is obtained through a reverse DNS lookup. Reverse DNS lookup occurs after the network has been initialized on a node and can take time to resolve. Other system services can start prior to this and detect the hostname as **localhost** or similar. You can avoid this by using DHCP to provide the hostname for each cluster node.

Additionally, setting the hostnames through DHCP can bypass any manual DNS record name configuration errors in environments that have a DNS split-horizon implementation.

1.3.4.2. Network connectivity requirements

You must configure the network connectivity between machines to allow OpenShift Container Platform cluster components to communicate. Each machine must be able to resolve the hostnames of all other machines in the cluster.

This section provides details about the ports that are required.

IMPORTANT

In connected OpenShift Container Platform environments, all nodes are required to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Table 1.3. Ports used for all-machine to all-machine communications

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves
	22623	The port handles traffic from the Machine Config Server and directs the traffic to the control plane machines.
UDP	**6081**	Geneve
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
	123	Network Time Protocol (NTP) on UDP port **123**. If an external NTP time server is configured, you must open UDP port **123**.
	TCP/UDP	**30000**-**32767**
Kubernetes node port	ESP	N/A

Table 1.4. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 1.5. Ports used for control plane machine to control plane machine communications

Protocol	Port	Description
TCP	**2379-2380**	etcd server and peer ports

1.3.4.3. NTP configuration for user-provisioned infrastructure

OpenShift Container Platform clusters are configured to use a public Network Time Protocol (NTP) server by default. If you want to use a local enterprise NTP server, or if your cluster is being deployed in a disconnected network, you can configure the cluster to use a specific time server. For more information, see the documentation for *Configuring chrony time service* .

If a DHCP server provides NTP server information, the chrony time service on the Red Hat Enterprise Linux CoreOS (RHCOS) machines read the information and can sync the clock with the NTP servers.

Additional resources

- Configuring chrony time service

1.3.5. User-provisioned DNS requirements

In OpenShift Container Platform deployments, you must ensure that cluster components meet certain DNS name resolution criteria for internal communication, certificate validation, and automated node discovery purposes.

The following is a list of required cluster components:

- The Kubernetes API
- The OpenShift Container Platform application wildcard
- The bootstrap, control plane, and compute machines

Reverse DNS resolution is also required for the Kubernetes API, the bootstrap machine, the control plane machines, and the compute machines.

DNS A/AAAA or CNAME records are used for name resolution and PTR records are used for reverse name resolution. The reverse records are important because Red Hat Enterprise Linux CoreOS (RHCOS) uses the reverse records to set the hostnames for all the nodes, unless the hostnames are provided by DHCP. Additionally, the reverse records are used to generate the certificate signing requests (CSR) that OpenShift Container Platform needs to operate.

NOTE

It is recommended to use a DHCP server to provide the hostnames to each cluster node. See the *DHCP recommendations for user-provisioned infrastructure* section for more information.

The following DNS records are required for a user-provisioned OpenShift Container Platform cluster and they must be in place before installation. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>.**.

Table 1.6. Required DNS records

Component	Record	Description
Kubernetes API	**api.<cluster_name>.<base_domain>.**	A DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the API load balancer. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	A DNS A/AAAA or CNAME record, and a DNS PTR record, to internally identify the API load balancer. These records must be resolvable from all the nodes within the cluster.  **IMPORTANT** The API server must be able to resolve the worker nodes by the hostnames that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	***.apps.<cluster_name>.<base_domain>.**	A wildcard DNS A/AAAA or CNAME record that refers to the application ingress load balancer. The application ingress load balancer targets the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster. For example, **console-openshift-console.apps.<cluster_name>.<base_domain>** is used as a wildcard route to the OpenShift Container Platform console.
Bootstrap machine	**bootstrap.<cluster_name>.<base_domain>.**	A DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Control plane machines	**<control_plane><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the control plane nodes. These records must be resolvable by the nodes within the cluster.
Compute machines	**<compute><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

NOTE

In OpenShift Container Platform 4.4 and later, you do not need to specify etcd host and SRV records in your DNS configuration.

TIP

You can use the **dig** command to verify name and reverse name resolution. See the section on *Validating DNS resolution for user-provisioned infrastructure* for detailed validation steps.

1.3.5.1. Example DNS configuration for user-provisioned clusters

Reference the example DNS configurations to understand how A and PTR record configuration samples meet the DNS requirements for deploying OpenShift Container Platform on user-provisioned infrastructure.

The DNS configuration examples provided here are for reference only and are not meant to provide advice for choosing one DNS solution over another.

In the examples, the cluster name is **ocp4** and the base domain is **example.com**.

The following example is a BIND zone file that shows sample DNS A records for name resolution in a user-provisioned cluster.

NOTE

In the example, the same load balancer is used for the Kubernetes API and application ingress traffic. In production scenarios, you can deploy the API and application ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.5
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
helper.ocp4.example.com. IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5
api-int.ocp4.example.com. IN A 192.168.1.5
;
*.apps.ocp4.example.com. IN A 192.168.1.5
;
bootstrap.ocp4.example.com. IN A 192.168.1.96
;
```

```
control-plane0.ocp4.example.com. IN A 192.168.1.97
control-plane1.ocp4.example.com. IN A 192.168.1.98
;
control-plane2.ocp4.example.com. IN A 192.168.1.99
;
compute0.ocp4.example.com. IN A 192.168.1.11
compute1.ocp4.example.com. IN A 192.168.1.7
;
;EOF
```

where:

api.ocp4.example.com.

Provides name resolution for the Kubernetes API. The record refers to the IP address of the API load balancer.

api-int.ocp4.example.com.

Provides name resolution for the Kubernetes API. The record refers to the IP address of the API load balancer and is used for internal cluster communications.

***.apps.ocp4.example.com.**

Provides name resolution for the wildcard routes. The record refers to the IP address of the application ingress load balancer. The application ingress load balancer targets the machines that run the Ingress Controller pods.

bootstrap.ocp4.example.com

Provides name resolution for the bootstrap machine.

control-plane0.ocp4.example.com

Provides name resolution for the control plane machines.

compute0.ocp4.example.com.

Provides name resolution for the compute machines.

The following example BIND zone file shows sample PTR records for reverse name resolution in a user-provisioned cluster:

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com.
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com.
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com.
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com.
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com.
;
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com.
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com.
```

```
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com.
;
;EOF
```

where:

api.ocp4.example.com.

Provides reverse DNS resolution for the Kubernetes API. The PTR record refers to the record name of the API load balancer.

api-int.ocp4.example.com.

Provides reverse DNS resolution for the Kubernetes API. The PTR record refers to the record name of the API load balancer and is used for internal cluster communications.

bootstrap.ocp4.example.com.

Provides reverse DNS resolution for the bootstrap machine.

control-plane0.ocp4.example.com.

Provides rebootstrap.ocp4.example.com.verse DNS resolution for the control plane machines.

compute0.ocp4.example.com.

Provides reverse DNS resolution for the compute machines.

NOTE

A PTR record is not required for the OpenShift Container Platform application wildcard.

1.3.6. Load balancing requirements for user-provisioned infrastructure

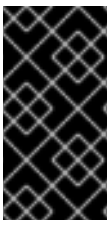
Before you install OpenShift Container Platform, you must provision the API and application Ingress load balancing infrastructure. In production scenarios, you can deploy the API and application Ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

NOTE

If you want to deploy the API and application Ingress load balancers with a Red Hat Enterprise Linux (RHEL) instance, you must purchase the RHEL subscription separately.

The load balancing infrastructure must meet the following requirements:

- API load balancer: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
  - Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.
  - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

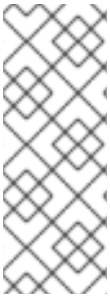


IMPORTANT

Do not configure session persistence for an API load balancer. Configuring session persistence for a Kubernetes API server might cause performance issues from excess application traffic for your OpenShift Container Platform cluster and the Kubernetes API that runs inside the cluster.

Configure the following ports on both the front and back of the API load balancers:

Port	Back-end machines (pool members)	Internal	External	Description
6443	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane. You must configure the **/readyz** endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server



NOTE

The load balancer must be configured to take a maximum of 30 seconds from the time the API server turns off the **/readyz** endpoint to the removal of the API server instance from the pool. Within the time frame after **/readyz** returns an error or becomes healthy, the endpoint must have been removed or added. Probing every 5 or 10 seconds, with two successful requests to become healthy and three to become unhealthy, are well-tested values.

- Application Ingress load balancer: Provides an ingress point for application traffic flowing in from outside the cluster. A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. Configure the following conditions:
  - Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.
  - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

TIP

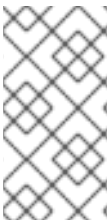
If the true IP address of the client can be seen by the application Ingress load balancer, enabling source IP-based session persistence can improve performance for applications that use end-to-end TLS encryption.

Configure the following ports on both the front and back of the load balancers:

Table 1.7. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description
443	The machines that run the Ingress Controller pods, compute, or worker, by default.	X	X	HTTPS traffic

Port	Back-end machines (pool members)	Internal	External	Description
80	The machines that run the Ingress Controller pods, compute, or worker, by default.	X	X	HTTP traffic



NOTE

If you are deploying a three-node cluster with zero compute nodes, the Ingress Controller pods run on the control plane nodes. In three-node cluster deployments, you must configure your application Ingress load balancer to route HTTP and HTTPS traffic to the control plane nodes.

1.3.6.1. Example load balancer configuration for user-provisioned clusters

Reference the example API and application Ingress load balancer configuration so that you can understand how to meet the load balancing requirements for user-provisioned clusters.

The sample is an **/etc/haproxy/haproxy.cfg** configuration for an HAProxy load balancer. The example is not meant to provide advice for choosing one load balancing solution over another.

TIP

If you are using HAProxy as a load balancer, you can check that the **haproxy** process is listening on ports **6443**, **22623**, **443**, and **80** by running **netstat -nltupe** on the HAProxy node.

In the example, the same load balancer is used for the Kubernetes API and application ingress traffic. In production scenarios, you can deploy the API and application ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

NOTE

If you are using HAProxy as a load balancer and SELinux is set to **enforcing**, you must ensure that the HAProxy service can bind to the configured TCP port by running **setsebool -P haproxy_connect_any=1**.

Sample API and application Ingress load balancer configuration

```
global
  log         127.0.0.1 local2
  pidfile     /var/run/haproxy.pid
  maxconn     4000
  daemon
defaults
  mode                    http
  log                     global
  option                  dontlognull
  option http-server-close
  option                  redispatch
```

```
  retries                 3
  timeout http-request    10s
  timeout queue           1m
  timeout connect         10s
  timeout client          1m
  timeout server          1m
  timeout http-keep-alive 10s
  timeout check           10s
  maxconn                 3000
listen api-server-6443
  bind *:6443
  mode tcp
  option  httpchk GET /readyz HTTP/1.0
  option  log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2 rise
3 backup
  server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall
2 rise 3
  server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall
2 rise 3
  server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall
2 rise 3
listen machine-config-server-22623
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443
  bind *:443
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:443 check inter 1s
  server compute1 compute1.ocp4.example.com:443 check inter 1s
listen ingress-router-80
  bind *:80
  mode tcp
  balance source
  server compute0 compute0.ocp4.example.com:80 check inter 1s
  server compute1 compute1.ocp4.example.com:80 check inter 1s
```

where:

listen api-server-6443

Port **6443** handles the Kubernetes API traffic and points to the control plane machines.

server bootstrap bootstrap.ocp4.example.com

The bootstrap entries must be in place before the OpenShift Container Platform cluster installation and they must be removed after the bootstrap process is complete.

listen machine-config-server

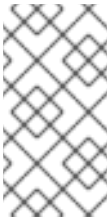
Port **22623** handles the machine config server traffic and points to the control plane machines.

listen ingress-router-443

Port **443** handles the HTTPS traffic and points to the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.

listen ingress-router-80

Port **80** handles the HTTP traffic and points to the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.



NOTE

If you are deploying a three-node cluster with zero compute nodes, the Ingress Controller pods run on the control plane nodes. In three-node cluster deployments, you must configure your application Ingress load balancer to route HTTP and HTTPS traffic to the control plane nodes.

1.4. PREPARING THE USER-PROVISIONED INFRASTRUCTURE

To ensure a successful deployment and meet cluster requirements in OpenShift Container Platform, prepare your user-provisioned infrastructure before starting the installation. Configuring your compute, network, and storage components in advance provides the stable foundation necessary for the installation program to function correctly.

This section provides details about the high-level steps required to set up your cluster infrastructure in preparation for an OpenShift Container Platform installation. This includes configuring IP networking and network connectivity for your cluster nodes, enabling the required ports through your firewall, and setting up the required DNS and load balancing infrastructure.

After preparation, your cluster infrastructure must meet the requirements outlined in the *Requirements for a cluster with user-provisioned infrastructure* section.

Prerequisites

- You have reviewed the OpenShift Container Platform 4.x Tested Integrations page.
- You have reviewed the infrastructure requirements detailed in the *Requirements for a cluster with user-provisioned infrastructure* section.

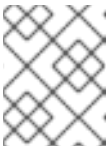
Procedure

1. If you are using DHCP to provide the IP networking configuration to your cluster nodes, configure your DHCP service.
   a. Add persistent IP addresses for the nodes to your DHCP server configuration. In your configuration, match the MAC address of the relevant network interface to the intended IP address for each node.
   b. When you use DHCP to configure IP addressing for the cluster machines, the machines also obtain the DNS server information through DHCP. Define the persistent DNS server address that is used by the cluster nodes through your DHCP server configuration.

NOTE

If you are not using a DHCP service, you must provide the IP networking configuration and the address of the DNS server to the nodes at RHCOS install time. These can be passed as boot arguments if you are installing from an ISO image. See the *Installing RHCOS and starting the OpenShift Container Platform bootstrap process* section for more information about static IP provisioning and advanced networking options.

c. Define the hostnames of your cluster nodes in your DHCP server configuration. See the *Setting the cluster node hostnames through DHCP* section for details about hostname considerations.

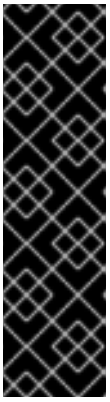


NOTE

If you are not using a DHCP service, the cluster nodes obtain their hostname through a reverse DNS lookup.

2. Ensure that your network infrastructure provides the required network connectivity between the cluster components. See the *Networking requirements for user-provisioned infrastructure* section for details about the requirements.

3. Configure your firewall to enable the ports required for the OpenShift Container Platform cluster components to communicate. See *Networking requirements for user-provisioned infrastructure* section for details about the ports that are required.



IMPORTANT

By default, port **1936** is accessible for an OpenShift Container Platform cluster, because each control plane node needs access to this port.

For ingress health check probes, the /**healthz/ready** endpoint is available on this port.

Avoid using the Ingress load balancer to expose this port, because doing so might result in the exposure of sensitive information, such as statistics and metrics, related to Ingress Controllers.

4. Setup the required DNS infrastructure for your cluster.

   a. Configure DNS name resolution for the Kubernetes API, the application wildcard, the bootstrap machine, the control plane machines, and the compute machines.

   b. Configure reverse DNS resolution for the Kubernetes API, the bootstrap machine, the control plane machines, and the compute machines.
   See the *User-provisioned DNS requirements* section for more information about the OpenShift Container Platform DNS requirements.

5. Validate your DNS configuration.

   a. From your installation node, run DNS lookups against the record names of the Kubernetes API, the wildcard routes, and the cluster nodes. Validate that the IP addresses in the responses correspond to the correct components.

b. From your installation node, run reverse DNS lookups against the IP addresses of the load balancer and the cluster nodes. Validate that the record names in the responses correspond to the correct components.
See the *Validating DNS resolution for user-provisioned infrastructure* section for detailed DNS validation steps.

6. Provision the required API and application ingress load balancing infrastructure. See the *Load balancing requirements for user-provisioned infrastructure* section for more information about the requirements.

NOTE

Some load balancing solutions require the DNS name resolution for the cluster nodes to be in place before the load balancing is initialized.

1.5. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE

To prevent network-related installation failures and ensure node connectivity in OpenShift Container Platform, validate your DNS configuration before deploying on user-provisioned infrastructure. This verification confirms that all required records resolve correctly, providing the stable foundation necessary for cluster communication.

IMPORTANT

The validation steps detailed in this section must succeed before you install your cluster.

Prerequisites

- You have configured the required DNS records for your user-provisioned infrastructure.

Procedure

1. From your installation node, run DNS lookups against the record names of the Kubernetes API, the wildcard routes, and the cluster nodes. Validate that the IP addresses contained in the responses correspond to the correct components.

   a. Perform a lookup against the Kubernetes API record name. Check that the result points to the IP address of the API load balancer:

   ```
   $ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain>
   ```

   Replace **<nameserver_ip>** with the IP address of the name server, **<cluster_name>** with your cluster name, and **<base_domain>** with your base domain name.

   Example output

   ```
   api.ocp4.example.com.  604800 IN A 192.168.1.5
   ```

   b. Perform a lookup against the Kubernetes internal API record name. Check that the result points to the IP address of the API load balancer:

   ```
   $ dig +noall +answer @<nameserver_ip> api-int.<cluster_name>.<base_domain>
   ```

Example output

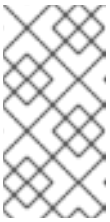
```
api-int.ocp4.example.com.  604800 IN A 192.168.1.5
```

c. Test an example ***.apps.<cluster_name>.<base_domain>*** DNS wildcard lookup. All of the application wildcard lookups must resolve to the IP address of the application ingress load balancer:

```
$ dig +noall +answer @<nameserver_ip> random.apps.<cluster_name>.<base_domain>
```

Example output

```
random.apps.ocp4.example.com.  604800 IN A 192.168.1.5
```



NOTE

In the example outputs, the same load balancer is used for the Kubernetes API and application ingress traffic. In production scenarios, you can deploy the API and application ingress load balancers separately so that you can scale the load balancer infrastructure for each in isolation.

You can replace **random** with another wildcard value. For example, you can query the route to the OpenShift Container Platform console:

```
$ dig +noall +answer @<nameserver_ip> console-openshift-console.apps.
<cluster_name>.<base_domain>
```

Example output

```
console-openshift-console.apps.ocp4.example.com. 604800 IN A 192.168.1.5
```

d. Run a lookup against the bootstrap DNS record name. Check that the result points to the IP address of the bootstrap node:

```
$ dig +noall +answer @<nameserver_ip> bootstrap.<cluster_name>.<base_domain>
```

Example output

```
bootstrap.ocp4.example.com.  604800 IN A 192.168.1.96
```

e. Use this method to perform lookups against the DNS record names for the control plane and compute nodes. Check that the results correspond to the IP addresses of each node.

2. From your installation node, run reverse DNS lookups against the IP addresses of the load balancer and the cluster nodes. Validate that the record names contained in the responses correspond to the correct components.

a. Perform a reverse lookup against the IP address of the API load balancer. Check that the response includes the record names for the Kubernetes API and the Kubernetes internal API:

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.5
```

Example output

```
5.1.168.192.in-addr.arpa. 604800 IN PTR api-int.ocp4.example.com.
5.1.168.192.in-addr.arpa. 604800 IN PTR api.ocp4.example.com.
```

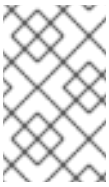
where:

api-int.ocp4.example.com

Specifies the record name for the Kubernetes internal API.

api.ocp4.example.com

Specifies the record name for the Kubernetes API.



NOTE

A PTR record is not required for the OpenShift Container Platform application wildcard. No validation step is needed for reverse DNS resolution against the IP address of the application ingress load balancer.

b. Perform a reverse lookup against the IP address of the bootstrap node. Check that the result points to the DNS record name of the bootstrap node:

```
$ dig +noall +answer @<nameserver_ip> -x 192.168.1.96
```

Example output

```
96.1.168.192.in-addr.arpa. 604800 IN PTR bootstrap.ocp4.example.com.
```

c. Use this method to perform reverse lookups against the IP addresses for the control plane and compute nodes. Check that the results correspond to the DNS record names of each node.

1.6. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS

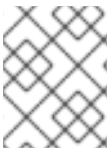
To enable secure, passwordless SSH access to your cluster nodes, provide an SSH public key during the OpenShift Container Platform installation. This ensures that the installation program automatically configures the Red Hat Enterprise Linux CoreOS (RHCOS) nodes for remote authentication through the **core** user.

The SSH public key gets added to the ~/**.ssh**/**authorized_keys** list for the **core** user on each node. After the key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The **./openshift-install gather** command also requires the SSH public key to be in place on the cluster nodes.

IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.



NOTE

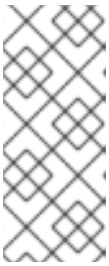
You must use a local key, not one that you configured with platform-specific approaches such as AWS key pairs.

Procedure

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name>
```

Specifies the path and file name, such as **~/.ssh/id_ed25519**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.



NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the **~/.ssh/id_ed25519.pub** public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the **./openshift-install gather** command.

NOTE

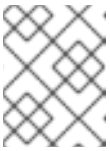
On some distributions, default SSH private key identities such as **~/.ssh/id_rsa** and **~/.ssh/id_dsa** are managed automatically.

a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

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

Example output

```
Agent pid 31874
```



NOTE

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

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

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

Specifies the path and file name for your SSH private key, such as **~/.ssh/id_ed25519**

Example output

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

Next steps

- When you install OpenShift Container Platform, provide the SSH public key to the installation program. If you install a cluster on infrastructure that you provision, you must provide the key to the installation program.

1.7. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

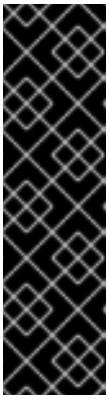
Procedure

1. Go to the Cluster Type page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

TIP

You can also download the binaries for a specific OpenShift Container Platform release .

2. Select your infrastructure provider from the **Run it yourself** section of the page.
3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.
4. Place the downloaded file in the directory where you want to store the installation configuration files.



IMPORTANT

- The installation program creates several files on the computer that you use to install your cluster. You must keep the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
- Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

```
$ tar -xvf openshift-install-linux.tar.gz
```

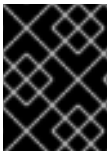
6. Download your installation pull secret from Red Hat OpenShift Cluster Manager . This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

TIP

Alternatively, you can retrieve the installation program from the Red Hat Customer Portal, where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

1.8. INSTALLING THE OPENSHIFT CLI ON LINUX

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on Linux.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.17. Download and install the new version of **oc**.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.17 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

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

To check your **PATH**, execute the following command:

```
$ echo $PATH
```

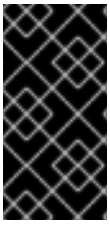
Verification

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

```
$ oc <command>
```

1.9. INSTALLING THE OPENSHIFT CLI ON WINDOWS

To manage your cluster and deploy applications from the command line, install OpenShift CLI (**oc**) binary on Windows.



IMPORTANT

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

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** list.
3. Click **Download Now** next to the **OpenShift v4.17 Windows Client** entry and save the file.
4. Extract the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH** variable.
   To check your **PATH** variable, open the command prompt and execute the following command:

```
C:\> path
```

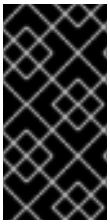
Verification

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

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

1.10. INSTALLING THE OPENSHIFT CLI ON MACOS

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on macOS.



IMPORTANT

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

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.17 macOS Clients** entry and save the file.

NOTE

For macOS arm64, choose the **OpenShift v4.17 macOS arm64 Client** entry.

5. Unpack and unzip the archive.
6. Move the **oc** binary to a directory on your **PATH** variable.
   To check your **PATH** variable, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

1.11. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE

To customise your OpenShift Container Platform deployment and meet specific network requirements, manually create the installation configuration file. This ensures that the installation program uses your tailored settings rather than default values during the setup process.

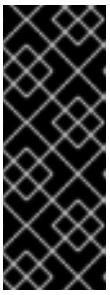
Prerequisites

- You have an SSH public key on your local machine for use with the installation program. You can use the key for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

1. Create an installation directory to store your required installation assets in:

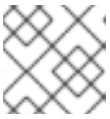
```
$ mkdir <installation_directory>
```



IMPORTANT

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

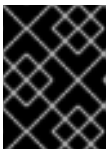
2. Customize the provided sample **install-config.yaml** file template and save the file in the **<installation_directory>**.



NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install many clusters.



IMPORTANT

Back up the **install-config.yaml** file now, because the installation process consumes the file in the next step.

1.11.1. Sample install-config.yaml file for other platforms

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

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: test
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
fips: false
pullSecret: '{"auths": ...}'
```

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

where:

baseDomain

Specifies the base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.

compute

Specifies the **compute** node configurations, which is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**.

controlPlane

Specifies the **controlPlane** node configurations, which is a single mapping. To meet the requirements of the different data structures, the first line of the **controlPlane** section must not. Only one control plane pool is used.

hyperthreading

Specifies whether to enable or disable simultaneous multithreading (SMT), or hyperthreading. By default, SMT is enabled to increase the performance of the cores in your machines. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.

NOTE

Simultaneous multithreading (SMT) is enabled by default. If SMT is not enabled in your BIOS settings, the **hyperthreading** parameter has no effect.

IMPORTANT

If you disable **hyperthreading**, whether in the BIOS or in the **install-config.yaml** file, ensure that your capacity planning accounts for the dramatically decreased machine performance.

compute.replicas

Specifies the number of compute machines that the cluster creates and manages for you on installer-provisioned installations. You must set this value to **0** when you install OpenShift Container Platform on user-provisioned infrastructure. Additionally for user-provisioned installations, you must manually deploy the compute machines before you finish installing the cluster.

NOTE

If you are installing a three-node cluster, do not deploy any compute machines when you install the Red Hat Enterprise Linux CoreOS (RHCOS) machines.

controlPlane.replicas

Specifies the number of control plane machines that you add to the cluster. Because the cluster uses these values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.

metadata.name

Specifies the cluster name that you specified in your DNS records.

clusterNetwork.cidr

Specifies a block of IP addresses from which pod IP addresses are allocated. This block must not overlap with existing physical networks. These IP addresses are used for the pod network. If you need to access the pods from an external network, you must configure load balancers and routers to manage the traffic.

NOTE

Class E CIDR range is reserved for a future use. To use the Class E CIDR range, you must ensure your networking environment accepts the IP addresses within the Class E CIDR range.

cidr.hostPrefix

Specifies the subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 (2^(32 - 23) - 2) pod IP addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.

networkType

Specifies the cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.

serviceNetwork

Specifies the IP address pool to use for service IP addresses. You can enter only one IP address pool. This block must not overlap with existing physical networks. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.

platform

Specifies the platform. You must set the platform to **none**. You cannot provide additional platform configuration variables for your platform.

IMPORTANT

Clusters that are installed with the platform type **none** are unable to use some features, such as managing compute machines with the Machine API. This limitation applies even if the compute machines that are attached to the cluster are installed on a platform that would normally support the feature. This parameter cannot be changed after installation.

fips

Specifies either enabling or disabling FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.

IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode .

When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.

pullSecret

Specifies the pull secret from Red Hat OpenShift Cluster Manager . This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

sshKey

Specifies the SSH public key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

NOTE

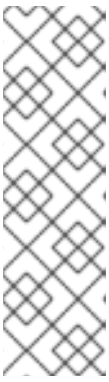
For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

1.11.2. Configuring the cluster-wide proxy during installation

To enable internet access in environments that deny direct connections, configure a cluster-wide proxy in the **install-config.yaml** file. This configuration ensures that the new OpenShift Container Platform cluster routes traffic through the specified HTTP or HTTPS proxy.

Prerequisites

- You have an existing **install-config.yaml** file.
- You have reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.



NOTE

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

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

Procedure

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

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port>
  httpsProxy: https://<username>:<pswd>@<ip>:<port>
  noProxy: example.com
additionalTrustBundle: |
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle>
# ...
```

where:

proxy.httpProxy

Specifies a proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

proxy.httpsProxy

Specifies a proxy URL to use for creating HTTPS connections outside the cluster.

proxy.noProxy

Specifies a comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use ***** to bypass the proxy for all destinations.

additionalTrustBundle

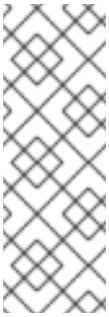
If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace to hold the additional CA certificates. If you provide **additionalTrustBundle** and at least one proxy setting, the **Proxy** object is configured to reference the **user-ca-bundle** config map in the **trustedCA** field. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges the contents specified for the **trustedCA** parameter with the RHCOS trust bundle. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

additionalTrustBundlePolicy

Specifies the policy that determines the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**. Optional parameter.

NOTE

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



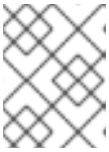
NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

+

```
$ ./openshift-install wait-for install-complete --log-level debug
```

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



NOTE

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

1.11.3. Configuring a three-node cluster

To create smaller, resource-efficient clusters for testing and production, deploy a bare-metal cluster with zero compute machines. This optional configuration uses only three control plane machines, optimizing infrastructure resources for administrators and developers.

In three-node OpenShift Container Platform environments, the three control plane machines are schedulable, which means that your application workloads are scheduled to run on them.

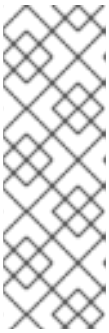
Prerequisites

- You have an existing **install-config.yaml** file.

Procedure

- Ensure that the number of compute replicas is set to **0** in your **install-config.yaml** file, as shown in the following **compute** stanza:

```
compute:
- name: worker
  platform: {}
  replicas: 0
# ...
```



NOTE

You must set the value of the **replicas** parameter for the compute machines to **0** when you install OpenShift Container Platform on user-provisioned infrastructure, regardless of the number of compute machines you are deploying. In installer-provisioned installations, the parameter controls the number of compute machines that the cluster creates and manages for you. This does not apply to user-provisioned installations, where the compute machines are deployed manually.

For three-node cluster installations, follow these next steps:

- If you are deploying a three-node cluster with zero compute nodes, the Ingress Controller pods run on the control plane nodes. In three-node cluster deployments, you must configure your application ingress load balancer to route HTTP and HTTPS traffic to the control plane nodes. See the *Load balancing requirements for user-provisioned infrastructure* section for more information.
- When you create the Kubernetes manifest files in the following procedure, ensure that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file is set to **true**. This enables your application workloads to run on the control plane nodes.
- Do not deploy any compute nodes when you create the Red Hat Enterprise Linux CoreOS (RHCOS) machines.

1.12. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES

To customize cluster definitions and manually start machines, generate the Kubernetes manifest and Ignition config files. These assets provide the necessary instructions to configure the cluster infrastructure according to your specific deployment requirements.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to configure the cluster machines.

IMPORTANT

- The Ignition config files that the OpenShift Container Platform installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

Prerequisites

- You obtained the OpenShift Container Platform installation program.
- You created the **install-config.yaml** installation configuration file.

Procedure

1. Change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

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

where

<installation_directory>

Specifies the installation directory that contains the **install-config.yaml** file you created.

WARNING

If you are installing a three-node cluster, skip the following step to allow the control plane nodes to be schedulable.

+

IMPORTANT

When you configure control plane nodes from the default unschedulable to schedulable, additional subscriptions are required. This is because control plane nodes then become compute nodes.

2. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.

   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.

   c. Save and exit the file.

3. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

where:

<installation_directory>

Specifies the same installation directory.
Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory. The **kubeadmin-password** and **kubeconfig** files are created in the **./<installation_directory>/auth** directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

1.13. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS

To install OpenShift Container Platform on bare-metal infrastructure that you provision, install Red Hat Enterprise Linux CoreOS (RHCOS) by using the generated Ignition config files. Providing these files ensures the bootstrap process begins automatically after the machines reboot.

If you have configured suitable networking, DNS, and load balancing infrastructure, the OpenShift Container Platform bootstrap process begins automatically after the RHCOS machines have rebooted.

To install RHCOS on the machines, follow either the steps to use an ISO image or network PXE booting.

NOTE

The compute node deployment steps included in this installation document are RHCOS-specific. If you choose instead to deploy RHEL-based compute nodes, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Only RHEL 8 compute machines are supported.

You can configure RHCOS during ISO and PXE installations by using the following methods:

- Kernel arguments: You can use kernel arguments to provide installation-specific information. For example, you can specify the locations of the RHCOS installation files that you uploaded to your HTTP server and the location of the Ignition config file for the type of node you are installing. For a PXE installation, you can use the **APPEND** parameter to pass the arguments to the kernel of the live installer. For an ISO installation, you can interrupt the live installation boot process to add the kernel arguments. In both installation cases, you can use special **coreos.inst.*** arguments to direct the live installer, as well as standard installation boot arguments for turning standard kernel services on or off.

- Ignition configs: OpenShift Container Platform Ignition config files (***.ign**) are specific to the type of node you are installing. You pass the location of a bootstrap, control plane, or compute node Ignition config file during the RHCOS installation so that it takes effect on first boot. In special cases, you can create a separate, limited Ignition config to pass to the live system. That Ignition config could do a certain set of tasks, such as reporting success to a provisioning system after completing installation. This special Ignition config is consumed by the **coreos-installer** to be applied on first boot of the installed system. Do not provide the standard control plane and compute node Ignition configs to the live ISO directly.

- **coreos-installer**: You can boot the live ISO installer to a shell prompt, which allows you to prepare the permanent system in a variety of ways before first boot. In particular, you can run the **coreos-installer** command to identify various artifacts to include, work with disk partitions, and set up networking. In some cases, you can configure features on the live system and copy them to the installed system.

NOTE

As of version **0.17.0-3**, **coreos-installer** requires RHEL 9 or later to run the program. You can still use older versions of **coreos-installer** to customize RHCOS artifacts of newer OpenShift Container Platform releases and install metal images to disk. You can download older versions of the **coreos-installer** binary from the **coreos-installer** image mirror page.

Whether to use an ISO or PXE install depends on your situation. A PXE install requires an available DHCP service and more preparation, but can make the installation process more automated. An ISO install is a more manual process and can be inconvenient if you are setting up more than a few machines.

1.13.1. Installing RHCOS by using an ISO image

To provision physical or virtual machines, install RHCOS by using a bootable ISO image. By using this method, you can deploy the operating system directly from local media or a virtual drive.

Prerequisites

- You have created the Ignition config files for your cluster.
- You have configured a suitable network, DNS, and load balancing infrastructure.
- You have an HTTP server that can be accessed from your computer, and from the machines that you create.
- You have reviewed the *Advanced RHCOS installation configuration* section for different ways to configure features, such as networking and disk partitioning.

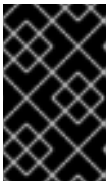
Procedure

1. Obtain the SHA512 digest for each of your Ignition config files. For example, you can use the following on a system running Linux to get the SHA512 digest for your **bootstrap.ign** Ignition config file:

```
$ sha512sum <installation_directory>/bootstrap.ign
```

   The digests are provided to the **coreos-installer** in a later step to validate the authenticity of the Ignition config files on the cluster nodes.

2. Upload the bootstrap, control plane, and compute node Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

You can add or change configuration settings in your Ignition configs before saving them to your HTTP server. If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

3. From the installation host, validate that the Ignition config files are available on the URLs. The following example gets the Ignition config file for the bootstrap node:

```
$ curl -k http://<HTTP_server>/bootstrap.ign
```

- <HTTP_server>: Replace **bootstrap.ign** with **master.ign** or **worker.ign** in the command to validate that the Ignition config files for the control plane and compute nodes are also available.

  Example output

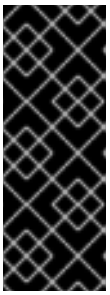
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

4. Although it is possible to obtain the RHCOS images that are required for your preferred method of installing operating system instances from the RHCOS image mirror page, the recommended way to obtain the correct version of your RHCOS images are from the output of **openshift-install** command:

```
$ openshift-install coreos print-stream-json | grep '\.iso[^.]'
```

Example output

```
"location": "<url>/art/storage/releases/rhcos-4.17-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.17-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.17-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.17/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```



IMPORTANT

The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Use the image versions that match your OpenShift Container Platform version if they are available. Use only ISO images for this procedure. RHCOS qcow2 images are not supported for this installation type.

ISO file names resemble the following example:

rhcos-<version>-live.<architecture>.iso

5. Use the ISO to start the RHCOS installation. Use one of the following installation options:
   - Burn the ISO image to a disk and boot it directly.
   - Use ISO redirection by using a lights-out management (LOM) interface.
6. Boot the RHCOS ISO image without specifying any options or interrupting the live boot sequence. Wait for the installer to boot into a shell prompt in the RHCOS live environment.

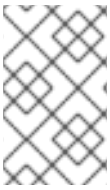
NOTE

It is possible to interrupt the RHCOS installation boot process to add kernel arguments. However, for this ISO procedure you should use the **coreos-installer** command as outlined in the following steps, instead of adding kernel arguments.

7. Run the **coreos-installer** command and specify the options that meet your installation requirements. At a minimum, you must specify the URL that points to the Ignition config file for the node type, and the device that you are installing to:

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device>
\
--ignition-hash=sha512-<digest>
```

- **<device>**: You must run the **coreos-installer** command by using **sudo**, because the **core** user does not have the required root privileges to perform the installation.
- **<digest>**: The **--ignition-hash** option is required when the Ignition config file is obtained through an HTTP URL to validate the authenticity of the Ignition config file on the cluster node. **<digest>** is the Ignition config file SHA512 digest obtained in a preceding step.



NOTE

If you want to provide your Ignition config files through an HTTPS server that uses TLS, you can add the internal certificate authority (CA) to the system trust store before running **coreos-installer**.

+ The following example initializes a bootstrap node installation to the **/dev/sda** device. The Ignition config file for the bootstrap node is obtained from an HTTP web server with the IP address 192.168.1.2:

+

1. Monitor the progress of the RHCOS installation on the console of the machine.

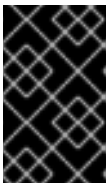
IMPORTANT

2. After RHCOS installs, you must reboot the system. During the system reboot, it applies the Ignition config file that you specified.
3. Check the console output to verify that Ignition ran.

Example command

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

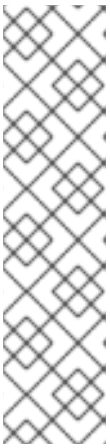
4. Continue to create the other machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, also create at least two compute machines before you install OpenShift Container Platform.

If the required network, DNS, and load balancer infrastructure are in place, the OpenShift Container Platform bootstrap process begins automatically after the RHCOS nodes have rebooted.



NOTE

RHCOS nodes do not include a default password for the **core** user. You can access the nodes by running **ssh core@<node>.<cluster_name>.<base_domain>** as a user with access to the SSH private key that is paired to the public key that you specified in your **install_config.yaml** file. OpenShift Container Platform 4 cluster nodes running RHCOS are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. However, when investigating installation issues, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on a target node, SSH access might be required for debugging or disaster recovery.

1.13.2. Installing RHCOS by using PXE or iPXE booting

You can use PXE or iPXE booting to install RHCOS on the machines.

Prerequisites

- You have created the Ignition config files for your cluster.
- You have configured suitable network, DNS and load balancing infrastructure.
- You have configured suitable PXE or iPXE infrastructure.
- You have an HTTP server that can be accessed from your computer, and from the machines that you create.
- You have reviewed the *Advanced RHCOS installation configuration* section for different ways to configure features, such as networking and disk partitioning.

Procedure

1. Upload the bootstrap, control plane, and compute node Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.

IMPORTANT

You can add or change configuration settings in your Ignition configs before saving them to your HTTP server. If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

2. From the installation host, validate that the Ignition config files are available on the URLs. The following example gets the Ignition config file for the bootstrap node:

```
$ curl -k http://<HTTP_server>/bootstrap.ign
```

- **<HTTP_server>**: Replace **bootstrap.ign** with **master.ign** or **worker.ign** in the command to validate that the Ignition config files for the control plane and compute nodes are also available.

Example output

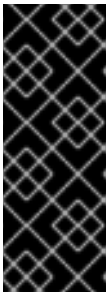
```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0{"ignition":
{"version":"3.2.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...
```

3. Although it is possible to obtain the RHCOS **kernel**, **initramfs** and **rootfs** files that are required for your preferred method of installing operating system instances from the RHCOS image mirror page, the recommended way to obtain the correct version of your RHCOS files are from the output of **openshift-install** command:

```
$ openshift-install coreos print-stream-json | grep -Eo '"https.*(kernel-|initramfs.|rootfs.)\w+(\.img)?"'
```

Example output

```
"<url>/art/storage/releases/rhcos-4.17-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.17-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.17-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.17-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.17-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.17-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.17-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.17-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.17-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.17/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.17/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.17/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```



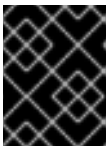
IMPORTANT

The RHCOS artifacts might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Only use the appropriate **kernel**, **initramfs**, and **rootfs** artifacts described below for this procedure. RHCOS QCOW2 images are not supported for this installation type.

The file names contain the OpenShift Container Platform version number. They resemble the following examples:

- **kernel**: **rhcos-<version>-live-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-live-initramfs.<architecture>.img**
- **rootfs**: **rhcos-<version>-live-rootfs.<architecture>.img**

4. Upload the **rootfs**, **kernel**, and **initramfs** files to your HTTP server.



IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.

6. Configure PXE or iPXE installation for the RHCOS images and begin the installation.

7. Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE (**x86_64**):

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture>
    APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
```

where:

kernel

Specify the location of the live **kernel** file that you uploaded to your HTTP server. The URL must be HTTP, TFTP, or FTP; HTTPS and NFS are not supported.

initrd=main

If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**. Specify the locations of the RHCOS files that you uploaded to your HTTP server. The **initrd** parameter value is the location of the **initramfs** file, the **coreos.live.rootfs_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file. You can also add more kernel arguments to the **APPEND** line to configure networking or other boot options.

NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? and "Enabling the serial console for PXE and ISO installation" in the "Advanced RHCOS installation configuration" section.

- For iPXE (**x86_64** + **aarch64**):

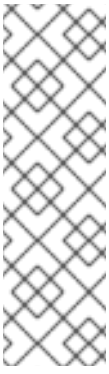
```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img
boot
```

kernel

Specify the locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd=main** argument is needed for booting on UEFI systems, the **coreos.live.rootfs_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file. If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

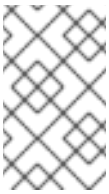
initrd

Specify the location of the **initramfs** file that you uploaded to your HTTP server.



NOTE

This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **kernel** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? and "Enabling the serial console for PXE and ISO installation" in the "Advanced RHCOS installation configuration" section.



NOTE

To network boot the CoreOS **kernel** on **aarch64** architecture, you need to use a version of iPXE build with the **IMAGE_GZIP** option enabled. See **IMAGE_GZIP** option in iPXE .

- For PXE (with UEFI and Grub as second stage) on **aarch64**:

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign
    initrd rhcos-<version>-live-initramfs.<architecture>.img
}
```

where:

coreos.live.rootfs_url

Specify the locations of the RHCOS files that you uploaded to your HTTP/TFTP server.

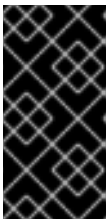
kernel

The **kernel** parameter value is the location of the **kernel** file on your TFTP server. The **coreos.live.rootfs_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file on your HTTP Server. If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

initrd rhcos

Specify the location of the **initramfs** file that you uploaded to your TFTP server.

8. Monitor the progress of the RHCOS installation on the console of the machine.



IMPORTANT

Be sure that the installation is successful on each node before commencing with the OpenShift Container Platform installation. Observing the installation process can also help to determine the cause of RHCOS installation issues that might arise.

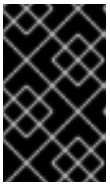
9. After RHCOS installs, the system reboots. During reboot, the system applies the Ignition config file that you specified.

10. Check the console output to verify that Ignition ran.

Example command

```
Ignition: ran on 2022/03/14 14:48:33 UTC (this boot)
Ignition: user-provided config was applied
```

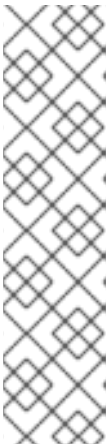
11. Continue to create the machines for your cluster.



IMPORTANT

You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, also create at least two compute machines before you install the cluster.

If the required network, DNS, and load balancer infrastructure are in place, the OpenShift Container Platform bootstrap process begins automatically after the RHCOS nodes have rebooted.



NOTE

RHCOS nodes do not include a default password for the **core** user. You can access the nodes by running **ssh core@<node>.<cluster_name>.<base_domain>** as a user with access to the SSH private key that is paired to the public key that you specified in your **install_config.yaml** file. OpenShift Container Platform 4 cluster nodes running RHCOS are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. However, when investigating installation issues, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on a target node, SSH access might be required for debugging or disaster recovery.

1.13.3. Advanced RHCOS installation configuration

To apply advanced configurations unavailable through default installation methods, manually provision Red Hat Enterprise Linux CoreOS (RHCOS) nodes for OpenShift Container Platform. This approach enables granular control over the node infrastructure to meet specific deployment requirements.

- Passing kernel arguments to the live installer
- Running **coreos-installer** manually from the live system
- Customizing a live ISO or PXE boot image

The advanced configuration topics for manual Red Hat Enterprise Linux CoreOS (RHCOS) installations detailed in this section relate to disk partitioning, networking, and configuring Ignition in different ways.

1.13.4. Networking and bonding options for ISO installations

You can configure advanced options so that you can modify the Red Hat Enterprise Linux CoreOS (RHCOS) manual installation process. The subsequent sections show examples of networking options for an ISO installation.

If you install RHCOS from an ISO image, you can add kernel arguments manually when you boot the image to configure networking for a node. If no networking arguments are specified, DHCP is activated in the initramfs when RHCOS detects that networking is required to fetch the Ignition config file.

IMPORTANT

When adding networking arguments manually, you must also add the **rd.neednet=1** kernel argument to bring the network up in the initramfs.

The following information provides examples for configuring networking and bonding on your RHCOS nodes for ISO installations. The examples describe how to use the **ip=**, **nameserver=**, and **bond=** kernel arguments.

NOTE

Ordering is important when adding the kernel arguments: **ip=**, **nameserver=**, and then **bond=**.

The networking options are passed to the **dracut** tool during system boot. For more information about the networking options supported by **dracut**, see **dracut.cmdline** manual page.

Additional resources

- **dracut.cmdline** manual page

1.13.4.1. Configuring DHCP or static IP addresses

You can configure an IP address by using either DHCP or an individual static IP address. If you set a static IP, you must then identify the DNS server IP address on each node.

The configuration examples in the procedure, update the IP addresses for the following components:

- The node's IP address to **10.10.10.2**
- The gateway address to **10.10.10.254**
- The netmask to **255.255.255.0**
- The hostname to **core0.example.com**
- The DNS server address to **4.4.4.41**
- The auto-configuration value to **none**. No auto-configuration is required when IP networking is configured statically.

Procedure

1. Enter a command like the following command to configure a static IP address:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
nameserver=4.4.4.41
```

2. Enter a command like the following command to configure a DHCP IP address:

```
ip=enp1s0:dhcp
```

NOTE

When you use DHCP to configure IP addressing for the RHCOS machines, the machines also obtain the DNS server information through DHCP. For DHCP-based deployments, you can define the DNS server address that is used by the RHCOS nodes through your DHCP server configuration.

3. If two or more network interfaces and only one interface exists, disable DHCP on a single interface. In the example, the **enp1s0** interface has a static networking configuration and DHCP is disabled for **enp2s0**, which is not used:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=::::core0.example.com:enp2s0:none
```

4. If you need to combine DHCP and static IP configurations on systems with multiple network interfaces, run the following example command:

```
ip=enp1s0:dhcp
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

1.13.4.2. Configuring an IP address without a static hostname

You can configure an IP address without assigning a static hostname. If a static hostname is not set by the user, the static hostname gets picked up and automatically set by a reverse DNS lookup.

The configuration examples in the procedure, update the IP addresses for the following components:

- The node's IP address to **10.10.10.2**
- The gateway address to **10.10.10.254**
- The netmask to **255.255.255.0**
- The DNS server address to **4.4.4.41**
- The auto-configuration value to **none**. No auto-configuration is required when IP networking is configured statically.

Procedure

- To configure an IP address without a static hostname, enter a command like the following command:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0::enp1s0:none
nameserver=4.4.4.41
```

1.13.4.3. Specifying multiple network interfaces and DNS servers

You can specify multiple network interfaces by setting multiple **ip=** entries. You can provide multiple DNS servers by adding a **nameserver=** entry for each server,

Procedure

- To specify multiple network interfaces for your interfaces, you can enter a command like the following command:

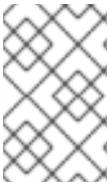
```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none
ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
```

- To provide multiple DNS servers by adding a **nameserver=** entry for each server, enter a command like the following command:

```
nameserver=1.1.1.1
nameserver=8.8.8.8
```

1.13.4.4. Configuring default gateway and route

As an optional task, you can configure routes to additional networks by setting an **rd.route=** value.



NOTE

When you configure one or multiple networks, one default gateway is required. If the additional network gateway is different from the primary network gateway, the default gateway must be the primary network gateway.

Procedure

- To configure the default gateway, enter the following command:

```
ip=::10.10.10.254::::
```

- To configure the route for an additional network, enter the following command:

```
rd.route=20.20.20.0/24:20.20.20.254:enp2s0
```

1.13.4.5. Configuring VLANs on individual interfaces

As an optional task, you can configure VLANs on individual interfaces by using the **vlan=** parameter.

Procedure

- To configure a VLAN on a network interface and use a static IP address, run the following command:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0.100:none
vlan=enp2s0.100:enp2s0
```

- To configure a VLAN on a network interface and to use DHCP, run the following command:

```
ip=enp2s0.100:dhcp
vlan=enp2s0.100:enp2s0
```

1.13.4.6. Bonding multiple network interfaces to a single interface

As an optional task, you can bond multiple network interfaces to a single interface by using the **bond=** option. By completing this task, you can eliminate a single point of failure for your network environment.

The following example demonstrates editing the **/etc/config/network** file and specifying the following syntax for bonding multiple network interfaces to a single interface:

```
bond=<name>[:<network_interfaces>][:<options>]
```

- **<name>**: Specifies the bonding device name, for example **bond0**.
- **<network_interfaces>**: Specifies a comma-separated list of physical (ethernet) interfaces, such as **em1,em2**.

- **<options>: Specifies a comma-separated list of bonding options. Enter the `modinfo bonding** command to see available options.

When you create a bonded interface using the **bond=** command, you must specify how the IP address is assigned and other information for the bonded interface.

Procedure

- To configure the bonded interface to use DHCP, edit the **/etc/config/network** file by setting the IP address for the bond to **dhcp**. For example:

```
bond=bond0:em1,em2:mode=active-backup
ip=bond0:dhcp
```

- To configure the bonded interface to use a static IP address, edit the **/etc/config/network** file entering the specific IP address you want and related information. For example:

```
bond=bond0:em1,em2:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

1.13.4.7. Bonding multiple SR-IOV network interfaces to a dual port NIC interface

You can bond multiple SR-IOV network interfaces to a dual port NIC interface by using the **bond=** option. This task provides high availability capabilities to your network by preventing a single physical port from becoming a single point of failure. Ensure you apply the procedure tasks to each node.

Procedure

1. Create the SR-IOV virtual functions (VFs) following the guidance in Managing SR-IOV devices. Follow the procedure in the "Attaching SR-IOV networking devices to virtual machines" section.

2. Create the bond, attach the desired VFs to the bond and set the bond link state up following the guidance in Configuring network bonding. Follow any of the described procedures to create the bond.
   The following examples illustrate the syntax you must use:

   - The syntax for configuring a bonded interface is **bond=<name>[:<network_interfaces>][:options]**.
     <name> is the bonding device name (**bond0**), **<network_interfaces>** represents the virtual functions (VFs) by their known name in the kernel and shown in the output of the **ip link** command(**eno1f0**, **eno2f0**), and *options* is a comma-separated list of bonding options. Enter **modinfo bonding** to see available options.

   - When you create a bonded interface using **bond=**, you must specify how the IP address is assigned and other information for the bonded interface.

     - To configure the bonded interface to use DHCP, set the bond's IP address to **dhcp**. For example:

```
bond=bond0:eno1f0,eno2f0:mode=active-backup
ip=bond0:dhcp
```

- To configure the bonded interface to use a static IP address, enter the specific IP address you want and related information. For example:

```
bond=bond0:eno1f0,eno2f0:mode=active-backup
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

3. Optional: You can use network teaming as an alternative to bonding by using the **team=** parameter.

- The syntax for configuring a team interface is: **team=name[:network_interfaces]** *name* is the team device name (**team0**) and *network_interfaces* represents a comma-separated list of physical (ethernet) interfaces (**em1, em2**).

NOTE

Teaming is planned to be deprecated when RHCOS switches to an upcoming version of RHEL. For more information, see this Red Hat Knowledgebase Article.

Use the following example to configure a network team:

```
team=team0:em1,em2
ip=team0:dhcp
```

1.13.4.8. coreos-installer and boot options for ISO and PXE installations

You can install RHCOS by running **coreos-installer install <options> <device>** at the command prompt, after booting into the RHCOS live environment from an ISO image.

The following table shows the subcommands, options, and arguments you can pass to the **coreos-installer** command.

Table 1.8. coreos-installer subcommands, command-line options, and arguments

coreos-installer install subcommand	
Subcommand	*Description*
$ coreos-installer install <options> <device>	Embed an Ignition config in an ISO image.
coreos-installer install subcommand options	
Option	*Description*
-u, --image-url <url>	Specify the image URL manually.
-f, --image-file <path>	Specify a local image file manually. Used for debugging.
-i, --ignition-file <path>	Embed an Ignition config from a file.

-I, --ignition-url <URL>	Embed an Ignition config from a URL.
--ignition-hash <digest>	Digest **type-value** of the Ignition config.
-p, --platform <name>	Override the Ignition platform ID for the installed system.
--console <spec>	Set the kernel and boot loader console for the installed system. For more information about the format of **<spec>**, see the Linux kernel serial console documentation.
--append-karg <arg>...	Append a default kernel argument to the installed system.
--delete-karg <arg>...	Delete a default kernel argument from the installed system.
-n, --copy-network	Copy the network configuration from the install environment.  **IMPORTANT** The **--copy-network** option only copies networking configuration found under **/etc/NetworkManager/system-connections**. In particular, it does not copy the system hostname.
--network-dir <path>	For use with **-n**. Default is **/etc/NetworkManager/system-connections/**.
--save-partlabel <lx>..	Save partitions with this label glob.
--save-partindex <id>...	Save partitions with this number or range.
--insecure	Skip RHCOS image signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Valid values are **x86_64** and **aarch64**.
--preserve-on-error	Do not clear partition table on error.
-h, --help	Print help information.

coreos-installer install subcommand argument	
Argument	*Description*
<device>	The destination device.
coreos-installer ISO subcommands	
Subcommand	*Description*
$ coreos-installer iso customize <options> <ISO_image>	Customize a RHCOS live ISO image.
coreos-installer iso reset <options> <ISO_image>	Restore a RHCOS live ISO image to default settings.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.
coreos-installer ISO customize subcommand options	
Option	*Description*
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and boot loader console for the destination system.
--dest-device <path>	Install and overwrite the specified destination device.
--dest-karg-append <arg>	Add a kernel argument to each boot of the destination system.
--dest-karg-delete <arg>	Delete a kernel argument from each boot of the destination system.
--network-keyfile <path>	Configure networking by using the specified NetworkManager keyfile for live and destination systems.
--ignition-ca <path>	Specify an additional TLS certificate authority to be trusted by Ignition.
--pre-install <path>	Run the specified script before installation.
--post-install <path>	Run the specified script after installation.

--installer-config <path>	Apply the specified installer configuration file.
--live-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
--live-karg-append <arg>	Add a kernel argument to each boot of the live environment.
--live-karg-delete <arg>	Delete a kernel argument from each boot of the live environment.
--live-karg-replace <k=o=n>	Replace a kernel argument in each boot of the live environment, in the form **key=old=new**.
-f, **--force**	Overwrite an existing Ignition config.
-o, **--output <path>**	Write the ISO to a new output file.
-h, **--help**	Print help information.
coreos-installer PXE subcommands	
Subcommand	*Description*
Note that not all of these options are accepted by all subcommands.	
coreos-installer pxe customize <options> <path>	Customize a RHCOS live PXE boot config.
coreos-installer pxe ignition wrap <options>	Wrap an Ignition config in an image.
coreos-installer pxe ignition unwrap <options> <image_name>	Show the wrapped Ignition config in an image.
coreos-installer PXE customize subcommand options	
Option	*Description*
Note that not all of these options are accepted by all subcommands.	
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and boot loader console for the destination system.
--dest-device <path>	Install and overwrite the specified destination device.

--network-keyfile <path>	Configure networking by using the specified NetworkManager keyfile for live and destination systems.
--ignition-ca <path>	Specify an additional TLS certificate authority to be trusted by Ignition.
--pre-install <path>	Run the specified script before installation.
post-install <path>	Run the specified script after installation.
--installer-config <path>	Apply the specified installer configuration file.
--live-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
-o, --output <path>	Write the initramfs to a new output file. NOTE This option is required for PXE environments.
-h, --help	Print help information.

You can automatically start **coreos-installer** options at boot time by passing **coreos.inst** boot arguments to the RHCOS live installer. These are provided in addition to the standard boot arguments.

- For ISO installations, the **coreos.inst** options can be added by interrupting the automatic boot at the boot loader menu. You can interrupt the automatic boot by pressing **TAB** while the **RHEL CoreOS (Live)** menu option is highlighted.
- For PXE or iPXE installations, the **coreos.inst** options must be added to the **APPEND** line before the RHCOS live installer is booted.

The following table shows the RHCOS live installer **coreos.inst** boot options for ISO and PXE installations.

Table 1.9. coreos.inst boot options

Argument	Description

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to.  **NOTE** It is recommended to use the full path, such as **/dev/sda**, although **sda** is allowed.
coreos.inst.ignition_url	Optional: The URL of the Ignition config to embed into the installed system. If no URL is specified, no Ignition config is embedded. Only HTTP and HTTPS protocols are supported.
coreos.inst.save_partlabel	Optional: Comma-separated labels of partitions to preserve during the install. Glob-style wildcards are permitted. The specified partitions do not need to exist.
coreos.inst.save_partindex	Optional: Comma-separated indexes of partitions to preserve during the install. Ranges **m-n** are permitted, and either **m** or **n** can be omitted. The specified partitions do not need to exist.
coreos.inst.insecure	Optional: Permits the OS image that is specified by **coreos.inst.image_url** to be unsigned.
coreos.inst.image_url	Optional: Download and install the specified RHCOS image. ● This argument should not be used in production environments and is intended for debugging purposes only. ● While this argument can be used to install a version of RHCOS that does not match the live media, it is recommended that you instead use the media that matches the version you want to install. ● If you are using **coreos.inst.image_url**, you must also use **coreos.inst.insecure**. This is because the bare-metal media are not GPG-signed for OpenShift Container Platform. ● Only HTTP and HTTPS protocols are supported.

Argument	Description
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. After the install finishes, you will receive a prompt that allows you to inspect what is happening during installation. This argument should not be used in production environments and is intended for debugging purposes only.
coreos.inst.platform_id	Optional: The Ignition platform ID of the platform the RHCOS image is being installed on. Default is **metal**. This option determines whether or not to request an Ignition config from the cloud provider, such as VMware. For example: **coreos.inst.platform_id=vmware**.
ignition.config.url	Optional: The URL of the Ignition config for the live boot. For example, this can be used to customize how **coreos-installer** is invoked, or to run code before or after the installation. This is different from **coreos.inst.ignition_url**, which is the Ignition config for the installed system.

1.14. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE

To install OpenShift Container Platform, use Ignition configuration files to initialize the bootstrap process after the cluster nodes boot into RHCOS. You must wait for this process to complete to ensure the cluster is fully installed.

Prerequisites

- You have created the Ignition config files for your cluster.
- You have configured suitable network, DNS, and load balancing infrastructure.
- You have obtained the installation program and generated the Ignition config files for your cluster.
- You installed RHCOS on your cluster machines and provided the Ignition config files that the OpenShift Container Platform installation program generated.
- Your machines have direct internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \
    --log-level=info
```

where:

<installation_directory>

Specifies the path to the directory that stores the installation files.

--log-level=info

Specifies **warn**, **debug**, or **error** instead of **info** to view different installation details.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.30.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After the bootstrap process is complete, remove the bootstrap machine from the load balancer.

IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the bootstrap machine itself.

1.15. LOGGING IN TO THE CLUSTER BY USING THE CLI

To log in to your cluster as the default system user, export the **kubeconfig** file. This configuration enables the CLI to authenticate and connect to the specific API server created during OpenShift Container Platform installation.

The **kubeconfig** file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

Procedure

1. Export the **kubeadmin** credentials by running the following command:

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

where:

<installation_directory>

Specifies the path to the directory that stores the installation files.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

Example output

```
system:admin
```

1.16. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

To add machines to a cluster, verify the status of the certificate signing requests (CSRs) generated for each machine. If manual approval is required, approve the client requests first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.30.3
master-1  Ready     master  63m  v1.30.3
master-2  Ready     master  64m  v1.30.3
```

The output lists all of the machines that you created.

NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name>
```

where:

<csr_name>

Specifies the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

   - To approve them individually, run the following command for each valid CSR:

     ```
     $ oc adm certificate approve <csr_name>
     ```

     where:

     <csr_name>

     Specifies the name of a CSR from the list of current CSRs.

   - To approve all pending CSRs, run the following command:

     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS   ROLES   AGE  VERSION
   master-0  Ready    master  73m  v1.30.3
   master-1  Ready    master  73m  v1.30.3
   master-2  Ready    master  74m  v1.30.3
   worker-0  Ready    worker  11m  v1.30.3
   worker-1  Ready    worker  11m  v1.30.3
   ```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

1.17. INITIAL OPERATOR CONFIGURATION

To ensure all Operators become available, configure the required Operators immediately after the control plane initialises. This configuration is essential for stabilizing the cluster environment following the installation.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.17.0    True        False         False      19m
baremetal                                  4.17.0    True        False         False      37m
cloud-credential                           4.17.0    True        False         False      40m
cluster-autoscaler                         4.17.0    True        False         False      37m
config-operator                            4.17.0    True        False         False      38m
console                                    4.17.0    True        False         False      26m
csi-snapshot-controller                    4.17.0    True        False         False      37m
dns                                        4.17.0    True        False         False      37m
etcd                                       4.17.0    True        False         False      36m
image-registry                             4.17.0    True        False         False      31m
ingress                                    4.17.0    True        False         False      30m
insights                                   4.17.0    True        False         False      31m
kube-apiserver                             4.17.0    True        False         False      26m
kube-controller-manager                    4.17.0    True        False         False      36m
kube-scheduler                             4.17.0    True        False         False      36m
kube-storage-version-migrator              4.17.0    True        False         False      37m
machine-api                                4.17.0    True        False         False      29m
machine-approver                           4.17.0    True        False         False      37m
machine-config                             4.17.0    True        False         False      36m
marketplace                                4.17.0    True        False         False      37m
monitoring                                 4.17.0    True        False         False      29m
network                                    4.17.0    True        False         False      38m
node-tuning                                4.17.0    True        False         False      37m
openshift-apiserver                        4.17.0    True        False         False      32m
openshift-controller-manager               4.17.0    True        False         False      30m
openshift-samples                          4.17.0    True        False         False      32m
operator-lifecycle-manager                 4.17.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.17.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.17.0    True        False         False      32m
service-ca                                 4.17.0    True        False         False      38m
storage                                    4.17.0    True        False         False      37m
```

2. Configure the Operators that are not available.

1.17.1. Disabling the default OperatorHub catalog sources

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

1.17.2. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**. When this has completed, you must configure storage.

1.17.3. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Configure a persistent volume, which is required for production clusters. Where applicable, you can configure an empty directory as the storage location for non-production clusters.

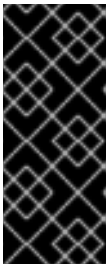
You can also allow the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

1.17.3.1. Configuring registry storage for bare metal and other manual installations

To ensure the registry is fully operational, configure the registry to use storage immediately after the cluster installation. This configuration is a mandatory step to enable the registry to store data.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have a cluster that uses manually-provisioned Red Hat Enterprise Linux CoreOS (RHCOS) nodes, such as bare metal.
- You have provisioned persistent storage for your cluster, such as Red Hat OpenShift Data Foundation.



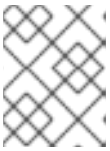
IMPORTANT

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. **ReadWriteOnce** access also requires that the registry uses the **Recreate** rollout strategy. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- You must have a system with at least 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.



NOTE

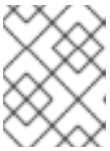
When you use shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

Example output

```
No resources found in openshift-image-registry namespace
```



NOTE

If you do have a registry pod in your output, you do not need to continue with this procedure.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

Example output

```
NAME             VERSION        AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.17           True        False         False      6h50m
```

5. Ensure that your registry is set to managed to enable building and pushing of images.

   - Run:

     ```
     $ oc edit configs.imageregistry/cluster
     ```

     Then, change the line

     ```
     managementState: Removed
     ```

     to

     ```
     managementState: Managed
     ```

1.17.3.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

- To set the image registry storage to an empty directory:

  ```
  $ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
  ```

  WARNING

  Configure this option only for non-production clusters.

  If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

  ```
  Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
  ```

  Wait a few minutes and run the command again.

1.17.3.3. Configuring block registry storage for bare metal

To allow the image registry to use block storage types during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.

IMPORTANT

Block storage volumes, or block persistent volumes, are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

If you choose to use a block storage volume with the image registry, you must use a filesystem persistent volume claim (PVC).

Procedure

1. Enter the following command to set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy, and runs with only one (**1**) replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.

   a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage
  namespace: openshift-image-registry
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

   where:

   metadata.name

   Specifies a unique name that represents the **PersistentVolumeClaim** object.

   metadata.namespace

   Specifies the **namespace** for the **PersistentVolumeClaim** object, which is **openshift-image-registry**.

   spec.accessModes

   Specifies the access mode of the persistent volume claim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.

   spec.resources.requests.storage

   The size of the persistent volume claim.

   b. Enter the following command to create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Enter the following command to edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:
  pvc:
    claim:
```

By creating a custom PVC, you can leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

1.18. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE

To finalize the installation on user-provisioned infrastructure, complete the cluster deployment after configuring the Operators. This ensures the cluster is fully operational on the infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.17.0    True        False         False      19m
baremetal                                  4.17.0    True        False         False      37m
cloud-credential                           4.17.0    True        False         False      40m
cluster-autoscaler                         4.17.0    True        False         False      37m
config-operator                            4.17.0    True        False         False      38m
console                                    4.17.0    True        False         False      26m
csi-snapshot-controller                    4.17.0    True        False         False      37m
dns                                        4.17.0    True        False         False      37m
etcd                                       4.17.0    True        False         False      36m
image-registry                             4.17.0    True        False         False      31m
ingress                                    4.17.0    True        False         False      30m
insights                                   4.17.0    True        False         False      31m
kube-apiserver                             4.17.0    True        False         False      26m
kube-controller-manager                    4.17.0    True        False         False      36m
kube-scheduler                             4.17.0    True        False         False      36m
kube-storage-version-migrator              4.17.0    True        False         False      37m
```

```
machine-api                                4.17.0    True        False         False      29m
machine-approver                           4.17.0    True        False         False      37m
machine-config                             4.17.0    True        False         False      36m
marketplace                                4.17.0    True        False         False      37m
monitoring                                 4.17.0    True        False         False      29m
network                                    4.17.0    True        False         False      38m
node-tuning                                4.17.0    True        False         False      37m
openshift-apiserver                        4.17.0    True        False         False      32m
openshift-controller-manager               4.17.0    True        False         False      30m
openshift-samples                          4.17.0    True        False         False      32m
operator-lifecycle-manager                 4.17.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.17.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.17.0    True        False         False      32m
service-ca                                 4.17.0    True        False         False      38m
storage                                    4.17.0    True        False         False      37m
```

Alternatively, the following command notifies you when all of the clusters are available. The command also retrieves and displays credentials:

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

where:

<installation_directory>

Specifies the path to the directory that you stored the installation files in.

Example output

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

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Confirm that the Kubernetes API server is communicating with the pods.

a. To view a list of all pods, use the following command:

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

Example output

```
NAMESPACE                       NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver             apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
Running    0          5m
```

b. View the logs for a pod that is listed in the output of the previous command by using the following command:

```
$ oc logs <pod_name> -n <namespace>
```

where:

<namespace>

Specifies the pod name and namespace, as shown in the output of an earlier command. If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

3. For an installation with Fibre Channel Protocol (FCP), additional steps are required to enable multipathing. Do not enable multipathing during installation.
See "Enabling multipathing with kernel arguments on RHCOS" in the *Postinstallation machine configuration tasks* documentation for more information.

1.19. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

To provide metrics about cluster health and the success of updates, the Telemetry service requires internet access. When connected, this service runs automatically by default and registers your cluster to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager,use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level. For more information about subscription watch, see "Data Gathered and Used by Red Hat's subscription services" in the *Additional resources* section.

Additional resources

- See About remote health monitoring for more information about the Telemetry service

1.20. NEXT STEPS

- Customize your cluster.
- If necessary, you can Remote health reporting.
- Set up your registry and configure registry storage .