OpenShift Container Platform 4.6

Installing on bare metal

Installing OpenShift Container Platform bare metal clusters

Last Updated: 2021-03-28

OpenShift Container Platform 4.6 Installing on bare metal

Installing OpenShift Container Platform bare metal clusters

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for installing OpenShift Container Platform clusters on bare metal infrastructure.

Table of Contents

CHAPTER 1. INSTALLING ON BARE METAL 5
1.1. INSTALLING A CLUSTER ON BARE METAL 5
1.1.1. Prerequisites 5
1.1.2. Internet and Telemetry access for OpenShift Container Platform 5
1.1.3. Machine requirements for a cluster with user-provisioned infrastructure 6
1.1.3.1. Required machines 6
1.1.3.2. Network connectivity requirements 6
1.1.3.3. Minimum resource requirements 6
1.1.3.4. Certificate signing requests management 7
1.1.4. Creating the user-provisioned infrastructure 7
1.1.4.1. Networking requirements for user-provisioned infrastructure 7
Network topology requirements 8
Load balancers 9
NTP configuration 10
1.1.4.2. User-provisioned DNS requirements 10
1.1.5. Generating an SSH private key and adding it to the agent 13
1.1.6. Obtaining the installation program 14
1.1.7. Installing the OpenShift CLI by downloading the binary 15
1.1.7.1. Installing the OpenShift CLI on Linux 15
1.1.7.2. Installing the OpenShift CLI on Windows 16
1.1.7.3. Installing the OpenShift CLI on macOS 16
1.1.8. Manually creating the installation configuration file 16
1.1.8.1. Sample install-config.yaml file for bare metal 17
1.1.8.2. Configuring the cluster-wide proxy during installation 19
1.1.9. Configuring a three-node cluster 21
1.1.10. Creating the Kubernetes manifest and Ignition config files 21
1.1.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines 23
1.1.11.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image 23
1.1.11.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting 25
1.1.11.3. Advanced Red Hat Enterprise Linux CoreOS (RHCOS) installation configuration 27
1.1.11.3.1. Using advanced networking options for PXE and ISO installations 27
1.1.11.3.2. Disk partitioning 28
1.1.11.3.2.1. Creating a separate /var partition 28
1.1.11.3.2.2. Retaining existing partitions 30
1.1.11.3.3. Identifying Ignition configs 31
1.1.11.3.3.1. Embedding an Ignition config in the RHCOS ISO 32
1.1.11.3.4. Advanced RHCOS installation reference 32
Routing and bonding options at RHCOS boot prompt 33
coreos.inst boot options for ISO or PXE install 34
coreos-installer options for ISO install 36
1.1.12. Creating the cluster 38
1.1.13. Logging in to the cluster by using the CLI 39
1.1.14. Approving the certificate signing requests for your machines 39
1.1.15. Initial Operator configuration 42
1.1.15.1. Image registry removed during installation 43
1.1.15.2. Image registry storage configuration 44
1.1.15.2.1. Configuring registry storage for bare metal and other manual installations 44
1.1.15.2.2. Configuring storage for the image registry in non-production clusters 45
1.1.15.2.3. Configuring block registry storage 46
1.1.16. Completing installation on user-provisioned infrastructure 46
1.1.17. Next steps 48

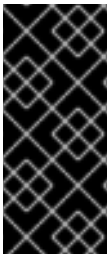
1.2. INSTALLING A CLUSTER ON BARE METAL WITH NETWORK CUSTOMIZATIONS 49
1.2.1. Prerequisites 49
1.2.2. Internet and Telemetry access for OpenShift Container Platform 49
1.2.3. Machine requirements for a cluster with user-provisioned infrastructure 49
1.2.3.1. Required machines 49
1.2.3.2. Network connectivity requirements 50
1.2.3.3. Minimum resource requirements 50
1.2.3.4. Certificate signing requests management 50
1.2.4. Creating the user-provisioned infrastructure 51
1.2.4.1. Networking requirements for user-provisioned infrastructure 51
Network topology requirements 52
Load balancers 52
NTP configuration 54
1.2.4.2. User-provisioned DNS requirements 54
1.2.5. Generating an SSH private key and adding it to the agent 57
1.2.6. Obtaining the installation program 58
1.2.7. Installing the OpenShift CLI by downloading the binary 59
1.2.7.1. Installing the OpenShift CLI on Linux 59
1.2.7.2. Installing the OpenShift CLI on Windows 59
1.2.7.3. Installing the OpenShift CLI on macOS 60
1.2.8. Manually creating the installation configuration file 60
1.2.8.1. Sample install-config.yaml file for bare metal 61
1.2.8.2. Network configuration parameters 63
1.2.9. Modifying advanced network configuration parameters 64
1.2.10. Cluster Network Operator configuration 65
1.2.10.1. Configuration parameters for the OpenShift SDN default CNI network provider 66
1.2.10.2. Configuration parameters for the OVN-Kubernetes default CNI network provider 67
1.2.10.3. Cluster Network Operator example configuration 67
1.2.11. Creating the Ignition config files 68
1.2.12. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines 69
1.2.12.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image 69
1.2.12.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting 71
1.2.12.3. Advanced Red Hat Enterprise Linux CoreOS (RHCOS) installation configuration 73
1.2.12.3.1. Using advanced networking options for PXE and ISO installations 73
1.2.12.3.2. Disk partitioning 74
1.2.12.3.2.1. Creating a separate /var partition 75
1.2.12.3.2.2. Retaining existing partitions 76
1.2.12.3.3. Identifying Ignition configs 77
1.2.12.3.3.1. Embedding an Ignition config in the RHCOS ISO 78
1.2.12.3.4. Advanced RHCOS installation reference 79
Routing and bonding options at RHCOS boot prompt 79
coreos.inst boot options for ISO or PXE install 81
coreos-installer options for ISO install 82
1.2.13. Creating the cluster 85
1.2.14. Logging in to the cluster by using the CLI 86
1.2.15. Approving the certificate signing requests for your machines 86
1.2.16. Initial Operator configuration 89
1.2.16.1. Image registry removed during installation 90
1.2.16.2. Image registry storage configuration 90
1.2.16.3. Configuring block registry storage 91
1.2.17. Completing installation on user-provisioned infrastructure 91
1.2.18. Next steps 93
1.3. INSTALLING A CLUSTER ON BARE METAL IN A RESTRICTED NETWORK 93

1.3.1. Prerequisites 94
1.3.2. About installations in restricted networks 94
1.3.2.1. Additional limits 95
1.3.3. Internet and Telemetry access for OpenShift Container Platform 95
1.3.4. Machine requirements for a cluster with user-provisioned infrastructure 95
1.3.4.1. Required machines 95
1.3.4.2. Network connectivity requirements 96
1.3.4.3. Minimum resource requirements 96
1.3.4.4. Certificate signing requests management 96
1.3.5. Creating the user-provisioned infrastructure 96
1.3.5.1. Networking requirements for user-provisioned infrastructure 97
Network topology requirements 98
Load balancers 98
NTP configuration 100
1.3.5.2. User-provisioned DNS requirements 100
1.3.6. Generating an SSH private key and adding it to the agent 103
1.3.7. Manually creating the installation configuration file 104
1.3.7.1. Sample install-config.yaml file for bare metal 105
1.3.7.2. Configuring the cluster-wide proxy during installation 107
1.3.8. Configuring a three-node cluster 109
1.3.9. Creating the Kubernetes manifest and Ignition config files 109
1.3.10. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines 110
1.3.10.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image 111
1.3.10.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting 112
1.3.10.3. Advanced Red Hat Enterprise Linux CoreOS (RHCOS) installation configuration 115
1.3.10.3.1. Using advanced networking options for PXE and ISO installations 115
1.3.10.3.2. Disk partitioning 116
1.3.10.3.2.1. Creating a separate /var partition 116
1.3.10.3.2.2. Retaining existing partitions 118
1.3.10.3.3. Identifying Ignition configs 119
1.3.10.3.3.1. Embedding an Ignition config in the RHCOS ISO 119
1.3.10.3.4. Advanced RHCOS installation reference 120
Routing and bonding options at RHCOS boot prompt 120
coreos.inst boot options for ISO or PXE install 122
coreos-installer options for ISO install 124
1.3.11. Creating the cluster 126
1.3.12. Logging in to the cluster by using the CLI 127
1.3.13. Approving the certificate signing requests for your machines 127
1.3.14. Initial Operator configuration 130
1.3.14.1. Image registry storage configuration 131
1.3.14.1.1. Changing the image registry's management state 131
1.3.14.1.2. Configuring registry storage for bare metal and other manual installations 132
1.3.14.1.3. Configuring storage for the image registry in non-production clusters 133
1.3.14.1.4. Configuring block registry storage 134
1.3.15. Completing installation on user-provisioned infrastructure 134
1.3.16. Next steps 136

CHAPTER 1. INSTALLING ON BARE METAL

1.1. INSTALLING A CLUSTER ON BARE METAL

In OpenShift Container Platform version 4.6, you can install a cluster on bare metal infrastructure that you provision.



IMPORTANT

While you might be able to follow this procedure to deploy a cluster on virtualized or cloud environments, you must be aware of additional considerations for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you attempt to install an OpenShift Container Platform cluster in such an environment.

1.1.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.
- If you use a firewall, you must configure it to allow the sites that your cluster requires access to.

NOTE

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

1.1.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the Red Hat OpenShift Cluster Manager (OCM) .

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

You must have Internet access to:

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

IMPORTANT

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

1.1.3. Machine requirements for a cluster with user-provisioned infrastructure

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

1.1.3.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines. If you are running a three-node cluster, running zero compute machines is supported. Running one compute machine is not supported.

NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

IMPORTANT

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

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See Red Hat Enterprise Linux technology capabilities and limits .

1.1.3.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files.

1.1.3.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU[1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

[1] 1 physical core provides 2 vCPUs when hyper-threading is enabled. 1 physical core provides 1 vCPU when hyper-threading is not enabled.

1.1.3.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

1.1.4. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the OpenShift Container Platform 4.x Tested Integrations page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

1.1.4.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

During the initial boot, the machines require either a DHCP server or that static IP addresses be set on each host in the cluster in order to establish a network connection, which allows them to download their Ignition config files.

It is recommended to use the DHCP server to manage the machines for the cluster long-term. Ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines.

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.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 1.1. All machines to all machines

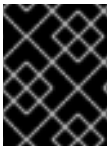
Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves
	10256	openshift-sdn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

Table 1.2. All machines to control plane

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server, peer, and metrics ports
	6443	Kubernetes API

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

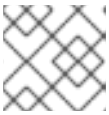
OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.



NOTE

Session persistence is not required for the API load balancer to function properly.

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

Table 1.3. API load balancer

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

NOTE

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

2. **Application Ingress load balancer**: Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
   - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

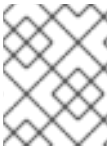
   Configure the following ports on both the front and back of the load balancers:

Table 1.4. Application Ingress load balancer

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

TIP

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



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

NTP configuration

An OpenShift Container Platform cluster that is installed in a restricted network is configured to use a public Network Time Protocol (NTP) server by default. To avoid clock skew, reconfigure the cluster to use a private NTP server instead. For more information, see the documentation for *Configuring chrony time service*.

Additional resources

- Configuring chrony time service

1.1.4.2. User-provisioned DNS requirements

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

The following DNS records are required for an OpenShift Container Platform cluster that uses user-provisioned infrastructure. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is

the cluster base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>.**.

Table 1.5. Required DNS records

Component	Record	Description
Kubernetes API	**api.<cluster_name>.<base_domain>.**	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster. **IMPORTANT** The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	***.apps.<cluster_name>.<base_domain>.**	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	**bootstrap.<cluster_name>.<base_domain>.**	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	**<master><n>.<cluster_name>.<base_domain>.**	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	**<worker><n>.<cluster_name>.<base_domain>.**	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 1.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

The following example BIND zone file shows sample PTR records for reverse name resolution.

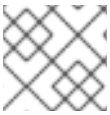
Example 1.2. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
```

```
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

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

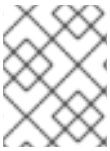
If you want to perform installation debugging or disaster recovery on your cluster, you must provide an SSH key to both your **ssh-agent** and the installation program. You can use this key to access the bootstrap machine in a public cluster to troubleshoot installation issues.



NOTE

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

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



NOTE

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

Procedure

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

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 Specify the path and file name, such as **~/.ssh/id_rsa**, of the new SSH key.

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

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

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

Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.1.6. Obtaining the installation program

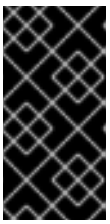
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

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

Procedure

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



IMPORTANT

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

IMPORTANT

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.

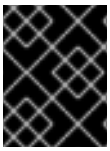
4. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

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

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

1.1.7. Installing the OpenShift CLI by downloading the binary

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



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.1.7.1. Installing the OpenShift CLI on Linux

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

Procedure

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

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

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.1.7.2. Installing the OpenShift CLI on Windows

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

Procedure

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

```
C:\> path
```

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

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

1.1.7.3. Installing the OpenShift CLI on macOS

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

Procedure

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.1.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

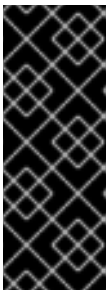
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

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

```
$ mkdir <installation_directory>
```



IMPORTANT

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

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

NOTE

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

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

IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

1.1.8.1. Sample install-config.yaml file for bare metal

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

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master (7)
  replicas: 3 (8)
metadata:
  name: test (9)
networking:
```

```
  clusterNetwork:
  - cidr: 10.128.0.0/14 10
    hostPrefix: 23 11
  networkType: OpenShiftSDN
  serviceNetwork: 12
  - 172.30.0.0/16
platform:
  none: {} 13
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

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

2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

3 6 7 Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.

IMPORTANT

If you disable simultaneous multithreading (SMT) using the **install-config.yaml** file or in your BIOS settings, ensure that your capacity planning accounts for the dramatically decreased machine performance. If you disable SMT in the BIOS, the **hyperthreading** parameter in the **install-config.yaml** file will have no effect.

4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.

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

9 The cluster name that you specified in your DNS records.

10 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.

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

12 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to

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

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

15 The pull secret that you obtained from the Pull Secret page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

16 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

NOTE

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

1.1.8.2. Configuring the cluster-wide proxy during installation

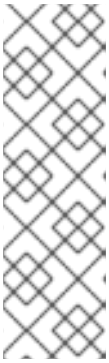
Production environments can deny direct access to the Internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

NOTE

For bare metal installations, if you do not assign node IP addresses from the range that is specified in the **networking.machineNetwork[].cidr** field in the **install-config.yaml** file, you must include them in the **proxy.noProxy** field.

Prerequisites

- You have an existing **install-config.yaml** file.
- You 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 Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

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

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

1. A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.

2. A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.

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

4. If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

NOTE

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

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

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

NOTE

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

1.1.9. Configuring a three-node cluster

You can optionally install and run three-node clusters in OpenShift Container Platform with no workers. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for development, production, and testing.

Procedure

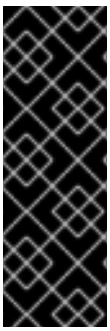
- Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

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

1.1.10. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

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



IMPORTANT

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

Prerequisites

- 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 installation program and generate the Kubernetes manifests for the cluster:

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

Example output

```
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: install_dir/manifests and install_dir/openshift
```

1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

WARNING

If you are running a three-node cluster, skip the following step to allow the masters to be schedulable.

1. 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.
2. 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> 1
```

1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

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

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

1.1.11. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. To create the machines, follow either the steps to use an ISO image or network PXE booting.

There are several methods of configuring RHCOS during ISO and PXE installations. These include:

- Kernel arguments: For a PXE install, you can **APPEND** arguments to the kernel of the live installer. For an ISO install, you can interrupt the live installation boot process to add kernel arguments. In both 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: You must generate an OpenShift Container Platform Ignition config file (***.ign**) for the type of node you are installing (worker, control plane, or bootstrap). You pass the location of the Ignition config to the installed system 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 installer and should not be used to include the standard **worker** and **master** Ignition configs.

- **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.

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.

NOTE

As of OpenShift Container Platform 4.6, the RHCOS ISO and other installation artifacts provide support for installation on disks with 4K sectors.

1.1.11.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

Before you install a cluster on infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

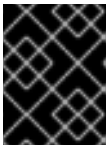
Prerequisites

- Obtain the Ignition config files for your cluster.

- Have access to an HTTP server that can be accessed from your computer, and from the machines that you create.

Procedure

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



IMPORTANT

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

2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the RHCOS image mirror page.

IMPORTANT

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

3. Use the ISO to start the RHCOS installation. Use one of the following installation options:
   - Burn the ISO image to a disk and boot it directly.
   - Use ISO redirection via a LOM interface.
4. Boot the ISO image. You can interrupt the installation boot process to add kernel arguments. However, for this ISO procedure you should use the **coreos-installer** command instead of adding kernel arguments. If you run the live installer without options or interruption, the installer boots up to a shell prompt on the live system, ready for you to install RHCOS to disk.
5. Review the *Advanced RHCOS installation reference* section for different ways of configuring features, such as networking and disk partitions, before running the **coreos-installer**.
6. Run the **coreos-installer** command. At a minimum, you must identify the Ignition config file location for your node type, and the location of the disk you are installing to. Here is an example:

```
$ sudo coreos-installer install \
    --ignition-url=https://host/worker.ign /dev/sda
```

7. After RHCOS installs, the system reboots. During the system reboot, it applies the Ignition config file that you specified.
8. 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, which is the default, also create at least two compute machines before you install the cluster.

1.1.11.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

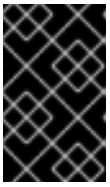
Before you install a cluster that uses manually-provisioned RHCOS nodes, such as bare metal, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Configure suitable PXE or iPXE infrastructure.
- Have access to an HTTP server that you can access from your computer.

Procedure

1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

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. Obtain the RHCOS **kernel**, **initramfs** and **rootfs** files from the Product Downloads page on the Red Hat customer portal or the RHCOS image mirror page.

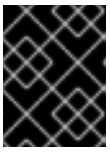
IMPORTANT

The RHCOS artifacts might not change with every release of OpenShift Container Platform. You must download artifacts 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**

3. 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.

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

5. Configure PXE or iPXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

   - For PXE:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    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 (2) (3)
```

   (1) 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.

   (2) If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

   (3) Specify locations of the RHCOS files that you uploaded to your HTTP 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?.

   - For iPXE:

```
kernel  http://<HTTP_server>/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 (1) (2)
initrd http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
boot
```

   (1) Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value is the location of the **initramfs** file. The **coreos.live.rootfs_url** parameter value is the

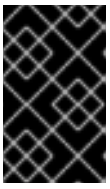
location of the **rootfs** file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.

2 If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

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?.

6. Continue to create the machines for your cluster.



IMPORTANT

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

1.1.11.3. Advanced Red Hat Enterprise Linux CoreOS (RHCOS) installation configuration

A key benefit for manually provisioning the Red Hat Enterprise Linux CoreOS (RHCOS) nodes for OpenShift Container Platform is to be able to do configuration that is not available through default OpenShift Container Platform installation methods. This section describes some of the configurations that you can do using techniques that include:

- Passing kernel arguments to the live installer
- Running **coreos-installer** manually from the live system
- Embedding Ignition configs in an ISO

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

1.1.11.3.1. Using advanced networking options for PXE and ISO installations

Networking for OpenShift Container Platform nodes uses DHCP by default to gather all necessary configuration settings. To set up static IP addresses or configure special settings, such as bonding, you can do one of the following:

- Pass special kernel parameters when you boot the live installer.
- Use a machine config to copy networking files to the installed system.
- Configure networking from a live installer shell prompt, then copy those settings to the installed system so that they take effect when the installed system first boots.

To configure a PXE or iPXE installation, use one of the following options:

- See the "Advanced RHCOS installation reference" tables.
- Use a machine config to copy networking files to the installed system.

To configure an ISO installation, use the following procedure.

Procedure

1. Boot the ISO installer.
2. From the live system shell prompt, configure networking for the live system using available RHEL tools, such as **nmcli** or **nmtui**.
3. Run the **coreos-installer** command to install the system, adding the **--copy-network** option to copy networking configuration. For example:

```
$ coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```

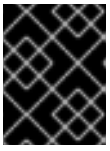
4. Reboot into the installed system.

1.1.11.3.2. Disk partitioning

In most cases, data partitions are originally created by installing RHCOS, rather than by installing another operating system. In such cases, the OpenShift Container Platform installer should be allowed to configure your disk partitions.

However, there are two cases where you might want to intervene to override the default partitioning when installing an OpenShift Container Platform node:

- Create separate partitions: For greenfield installations on an empty disk, you might want to add separate storage to a partition. This is officially supported for making **/var** or a subdirectory of **/var**, such as **/var/lib/etcd**, a separate partition, but not both.



IMPORTANT

Kubernetes supports only two filesystem partitions. If you add more than one partition to the original configuration, Kubernetes cannot monitor all of them.

- Retain existing partitions: For a brownfield installation where you are reinstalling OpenShift Container Platform on an existing node and want to retain data partitions installed from your previous operating system, there are both boot arguments and options to **coreos-installer** that allow you to retain existing data partitions.

1.1.11.3.2.1. Creating a separate /var partition

In general, disk partitioning for OpenShift Container Platform should be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.

- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.
- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a **/var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because **/var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate **/var** partition by creating a machine config that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. Create a **MachineConfig** object and add it to a file in the **openshift** directory. For example, name the file **98-var-partition.yaml**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This attaches storage to a separate **/var** directory.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      disks:
      - device: /dev/<device_name> 1
        partitions:
        - sizeMiB: <partition_size>
          startMiB: <partition_start_offset> 2
          label: var
      filesystems:
```

```
      - path: /var
        device: /dev/disk/by-partlabel/var
        format: xfs
    systemd:
      units:
        - name: var.mount
          enabled: true
          contents: |
            [Unit]
            Before=local-fs.target
            [Mount]
            Where=/var
            What=/dev/disk/by-partlabel/var
            [Install]
            WantedBy=local-fs.target
```

1. The storage device name of the disk that you want to partition.
2. When adding a data partition to the boot disk, a minimum value of 25000 mebibytes is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.

4. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Now you can use the Ignition config files as input to the ISO or PXE manual installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

1.1.11.3.2.2. Retaining existing partitions

For an ISO installation, you can add options to the **coreos-installer** command line that causes the installer to maintain one or more existing partitions. For a PXE installation, you can **APPEND coreos.inst.*** options to preserve partitions.

Saved partitions might be partitions from an existing OpenShift Container Platform system that has data partitions that you want to keep. Here are a few tips:

- If you save existing partitions, and those partitions do not leave enough space for RHCOS, installation will fail without damaging the saved partitions.
- Identify the disk partitions you want to keep either by partition label or by number.

For an ISO installation

This example preserves any partition in which the partition label begins with **data** (**data***):

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

The following example illustrates running the **coreos-installer** in a way that preserves the sixth (6) partition on the disk:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

This example preserves partitions 5 and higher:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

In the previous examples where partition saving is used, **coreos-installer** recreates the partition immediately.

For a PXE installation

This **APPEND** option preserves any partition in which the partition label begins with 'data' ('data*'):

```
coreos.inst.save_partlabel=data*
```

This **APPEND** option preserves partitions 5 and higher:

```
coreos.inst.save_partindex=5-
```

This **APPEND** option preserves partition 6:

```
coreos.inst.save_partindex=6
```

1.1.11.3.3. Identifying Ignition configs

When doing an RHCOS manual installation, there are two types of Ignition configs that you can provide, with different reasons for providing each one:

- **Permanent install Ignition config**: Every manual RHCOS installation needs to pass one of the Ignition config files generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, to carry out the installation.

  IMPORTANT

  It is not recommended to modify these files.

  For PXE installations, you pass the Ignition configs on the **APPEND** line using the **coreos.inst.ignition_url=** option. For ISO installations, after the ISO boots to the shell prompt, you identify the Ignition config on the **coreos-installer** command line with the **--ignition-url=** option. In both cases, only HTTP and HTTPS protocols are supported.

- **Live install Ignition config**: This type must be created manually and should be avoided if possible, as it is not supported by Red Hat. With this method, the Ignition config passes to the live install medium, runs immediately upon booting, and performs setup tasks before and/or after the RHCOS system installs to disk. This method should only be used for performing tasks that must be performed once and not applied again later, such as with advanced partitioning that cannot be done using a machine config.

For PXE or ISO boots, you can create the Ignition config and **APPEND** the **ignition.config.url=** option to identify the location of the Ignition config. You also need to append **ignition.firstboot ignition.platform.id=metal** or the **ignition.config.url** option will be ignored.

1.1.11.3.3.1. Embedding an Ignition config in the RHCOS ISO

You can embed a live install Ignition config directly in an RHCOS ISO image. When the ISO image is booted, the embedded config will be applied automatically.

Procedure

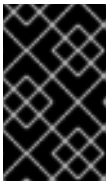
1. Download the **coreos-installer** binary from the following image mirror page: https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.
2. Retrieve the RHCOS ISO image and the Ignition config file, and copy them into an accessible directory, such as **/mnt**:

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. Run the following command to embed the Ignition config into the ISO:

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

You can now use that ISO to install RHCOS using the specified live install Ignition config.



IMPORTANT

Using **coreos-installer iso ignition embed** to embed a file generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, is unsupported and not recommended.

4. To show the contents of the embedded Ignition config and direct it into a file, run:

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

Example output

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. To remove the Ignition config and return the ISO to its pristine state so you can reuse it, run:

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

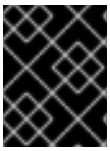
You can now embed another Ignition config into the ISO or use the ISO in its pristine state.

1.1.11.3.4. Advanced RHCOS installation reference

This section illustrates the networking configuration and other advanced options that allow you to modify the Red Hat Enterprise Linux CoreOS (RHCOS) manual installation process. The following tables

describe the kernel arguments and command-line options you can use with the RHCOS live installer and the **coreos-installer** command.

Routing and bonding options at RHCOS boot prompt
If you install RHCOS from an ISO image, you can add kernel arguments manually when you boot that image to configure the node's networking. If no networking arguments are used, the installation defaults to using DHCP.



IMPORTANT

When adding networking arguments, you must also add the **rd.neednet=1** kernel argument.

The following table describes how to use **ip=**, **nameserver=**, and **bond=** kernel arguments for live ISO installs.

NOTE

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

Table 1.6. Routing and bonding options for ISO

Description	Examples
To configure an IP address, either use DHCP (**ip=dhcp**) or set an individual static IP address (**ip=<host_ip>**). Then identify the DNS server IP address (**nameserver=<dns_ip>**) on each node. This example sets: • 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**	`ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none` `nameserver=4.4.4.41`
Specify multiple network interfaces by specifying multiple **ip=** entries.	`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`
Disable DHCP on a single interface, such as when there are two or more network interfaces and only one interface is being 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`

Description	Examples
You can combine DHCP and static IP configurations on systems with multiple network interfaces.	ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
You can provide multiple DNS servers by adding a **nameserver=** entry for each server.	nameserver=1.1.1.1 nameserver=8.8.8.8
Optional: Bonding multiple network interfaces to a single interface is supported using the **bond=** option. In these two examples: • The syntax for configuring a bonded interface is: **bond=name[:network_interfaces][:options]** • *name* is the bonding device name (**bond0**), *network_interfaces* represents a comma-separated list of physical (ethernet) interfaces (**em1,em2**), 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:em1,em2: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:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none

coreos.inst boot options for ISO or PXE install

While you can pass most standard installation boot arguments to the live installer, there are several arguments that are specific to the RHCOS live installer.

- For ISO, these options can be added by interrupting the RHCOS installer.
- For PXE or iPXE, these options must be added to the **APPEND** line before starting the PXE kernel. You cannot interrupt a live PXE install.

The following table shows the RHCOS live installer boot options for ISO and PXE installs.

Table 1.7. coreos.inst boot options

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to. It is recommended to use the full path, such as **/dev/sda**, although **sda** is allowed.

Argument	Description
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.
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.
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. Once 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**.

Argument	Description
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.

coreos-installer options for ISO install

You can also install RHCOS by invoking the **coreos-installer** command directly from the command line. The kernel arguments in the previous table provide a shortcut for automatically invoking **coreos-installer** at boot time, but you can pass similar arguments directly to **coreos-installer** when running it from a shell prompt.

The following table shows the options and subcommands you can pass to the **coreos-installer** command from a shell prompt during a live install.

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

Command-line options	
Option	**Description**
-u, **--image-url <url>**	Specify the image URL manually.
-f, **--image-file <path>**	Specify a local image file manually.
-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.
--append-karg <arg>…	Append the default kernel argument.
--delete-karg <arg>…	Delete the default kernel argument.
-n, **--copy-network**	Copy the network configuration from the install environment.
--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.

--offline	Force offline installation.
--insecure	Skip signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Default is **x86_64**.
--preserve-on-error	Do not clear partition table on error.
-h, **--help**	Print help information.

Command-line argument

Argument	Description
<device>	The destination device.

coreos-installer embedded Ignition commands

Command	Description
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	Embed an Ignition config in an ISO image.
coreos-installer iso ignition show <options> <ISO_image>	Show the embedded Ignition config from an ISO image.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.

coreos-installer ISO Ignition options

Option	Description
-f, **--force**	Overwrite an existing Ignition config.
-i, **--ignition-file <path>**	The Ignition config to be used. Default is **stdin**.
-o, **--output <path>**	Write the ISO to a new output file.
-h, **--help**	Print help information.

coreos-installer PXE Ignition commands

Command	Description
Note that not all of these options are accepted by all subcommands.	
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 ignition unwrap <options> <initrd_name>	Show the wrapped Ignition config in an **initrd** image.
coreos-installer PXE Ignition options	
Option	**Description**
-i, --ignition-file <path>	The Ignition config to be used. Default is **stdin**.
-o, --output <path>	Write the ISO to a new output file.
-h, --help	Print help information.

1.1.12. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

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

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.19.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

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

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

IMPORTANT

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

1.1.13. Logging in to the cluster by using the CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.1.14. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.19.0
master-1  Ready     master  63m  v1.19.0
master-2  Ready     master  64m  v1.19.0
worker-0  NotReady  worker  76s  v1.19.0
worker-1  NotReady  worker  70s  v1.19.0
```

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

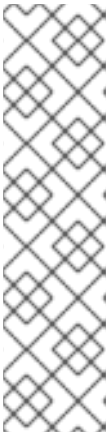
```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending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

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**.



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> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

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

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

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> 1
     ```

     1 **<csr_name>** is the name of a CSR from the list of current CSRs.

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

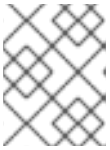
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

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

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```

   NOTE

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

Additional information

- For more information on CSRs, see Certificate Signing Requests.

1.1.15. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

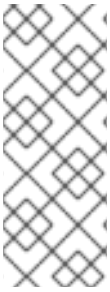
```
NAME                                       VERSION AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.6.0   True        False         False     3h56m
cloud-credential                           4.6.0   True        False         False     29h
cluster-autoscaler                         4.6.0   True        False         False     29h
config-operator                            4.6.0   True        False         False     6h39m
console                                    4.6.0   True        False         False     3h59m
csi-snapshot-controller                    4.6.0   True        False         False     4h12m
dns                                        4.6.0   True        False         False     4h15m
etcd                                       4.6.0   True        False         False     29h
image-registry                             4.6.0   True        False         False     3h59m
ingress                                    4.6.0   True        False         False     4h30m
insights                                   4.6.0   True        False         False     29h
kube-apiserver                             4.6.0   True        False         False     29h
kube-controller-manager                    4.6.0   True        False         False     29h
kube-scheduler                             4.6.0   True        False         False     29h
kube-storage-version-migrator              4.6.0   True        False         False     4h2m
machine-api                                4.6.0   True        False         False     29h
machine-approver                           4.6.0   True        False         False     6h34m
machine-config                             4.6.0   True        False         False     3h56m
marketplace                                4.6.0   True        False         False     4h2m
monitoring                                 4.6.0   True        False         False     6h31m
network                                    4.6.0   True        False         False     29h
node-tuning                                4.6.0   True        False         False     4h30m
openshift-apiserver                        4.6.0   True        False         False     3h56m
openshift-controller-manager               4.6.0   True        False         False     4h36m
openshift-samples                          4.6.0   True        False         False     4h30m
operator-lifecycle-manager                 4.6.0   True        False         False     29h
operator-lifecycle-manager-catalog         4.6.0   True        False         False     29h
operator-lifecycle-manager-packageserver   4.6.0   True        False         False     3h59m
service-ca                                 4.6.0   True        False         False     29h
storage                                    4.6.0   True        False         False     4h30m
```

2. Configure the Operators that are not available.

1.1.15.1. Image registry removed during installation

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

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing configs.imageregistry.operator.openshift.io."

1.1.15.2. Image registry storage configuration

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

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

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

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster that uses manually-provisioned Red Hat Enterprise Linux CoreOS (RHCOS) nodes, such as bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.

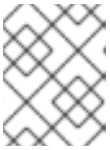
IMPORTANT

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

- Must have 100Gi capacity.

Procedure

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



NOTE

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

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

```
$ oc get pod -n openshift-image-registry
```

NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

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

Example output

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

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

4. Check the **clusteroperator** status:

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

5. 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
```

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

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

Procedure

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

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

WARNING

Configure this option for only non-production clusters.

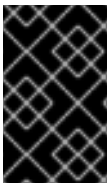
If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

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

Wait a few minutes and run the command again.

1.1.15.2.3. Configuring block registry storage

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



IMPORTANT

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

Procedure

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

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

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

3. Edit the registry configuration so that it references the correct PVC.

1.1.16. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

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

Procedure

1. Confirm that all the cluster components are online:

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

Example output

```
NAME                                       VERSION AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.6.0   True        False         False     3h56m
```

```
cloud-credential                           4.6.0   True        False       False      29h
cluster-autoscaler                         4.6.0   True        False       False      29h
config-operator                            4.6.0   True        False       False      6h39m
console                                    4.6.0   True        False       False      3h59m
csi-snapshot-controller                    4.6.0   True        False       False      4h12m
dns                                        4.6.0   True        False       False      4h15m
etcd                                       4.6.0   True        False       False      29h
image-registry                             4.6.0   True        False       False      3h59m
ingress                                    4.6.0   True        False       False      4h30m
insights                                   4.6.0   True        False       False      29h
kube-apiserver                             4.6.0   True        False       False      29h
kube-controller-manager                    4.6.0   True        False       False      29h
kube-scheduler                             4.6.0   True        False       False      29h
kube-storage-version-migrator              4.6.0   True        False       False      4h2m
machine-api                                4.6.0   True        False       False      29h
machine-approver                           4.6.0   True        False       False      6h34m
machine-config                             4.6.0   True        False       False      3h56m
marketplace                                4.6.0   True        False       False      4h2m
monitoring                                 4.6.0   True        False       False      6h31m
network                                    4.6.0   True        False       False      29h
node-tuning                                4.6.0   True        False       False      4h30m
openshift-apiserver                        4.6.0   True        False       False      3h56m
openshift-controller-manager               4.6.0   True        False       False      4h36m
openshift-samples                          4.6.0   True        False       False      4h30m
operator-lifecycle-manager                 4.6.0   True        False       False      29h
operator-lifecycle-manager-catalog         4.6.0   True        False       False      29h
operator-lifecycle-manager-packageserver   4.6.0   True        False       False      3h59m
service-ca                                 4.6.0   True        False       False      29h
storage                                    4.6.0   True        False       False      4h30m
```

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

2. Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

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

Example output

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

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

IMPORTANT

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

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

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

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

   Example output

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

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

   ```
   $ oc logs <pod_name> -n <namespace>
   ```
   (1)

   (1) Specify the pod name and namespace, as shown in the output of the previous command.

   If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

1.1.17. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting .
- Set up your registry and configure registry storage .

1.2. INSTALLING A CLUSTER ON BARE METAL WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.6, you can install a cluster on bare metal infrastructure that you provision with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

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

1.2.1. Prerequisites

- Review details about the OpenShift Container Platform installation and update processes.
- If you use a firewall, you must configure it to access Red Hat Insights .

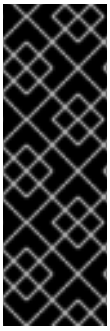
1.2.2. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the Red Hat OpenShift Cluster Manager (OCM) .

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

You must have Internet access to:

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



IMPORTANT

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

1.2.3. Machine requirements for a cluster with user-provisioned infrastructure

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

1.2.3.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines. If you are running a three-node cluster, running zero compute machines is supported. Running one compute machine is not supported.

NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

IMPORTANT

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

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See Red Hat Enterprise Linux technology capabilities and limits .

1.2.3.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files.

1.2.3.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU[1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

[1] 1 physical core provides 2 vCPUs when hyper-threading is enabled. 1 physical core provides 1 vCPU when hyper-threading is not enabled.

1.2.3.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure

that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

1.2.4. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the OpenShift Container Platform 4.x Tested Integrations page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

1.2.4.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

During the initial boot, the machines require either a DHCP server or that static IP addresses be set on each host in the cluster in order to establish a network connection, which allows them to download their Ignition config files.

It is recommended to use the DHCP server to manage the machines for the cluster long-term. Ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines.

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.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 1.9. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests

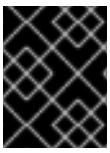
Protocol	Port	Description
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	**4789**	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.
TCP/UDP	**30000**-**32767**	Kubernetes node port

Table 1.10. All machines to control plane

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server, peer, and metrics ports
	6443	Kubernetes API

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

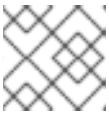
OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:
   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.

- A stateless load balancing algorithm. The options vary based on the load balancer implementation.



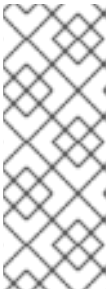
NOTE

Session persistence is not required for the API load balancer to function properly.

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

Table 1.11. API load balancer

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



NOTE

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

2. **Application Ingress load balancer**: Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.

   - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

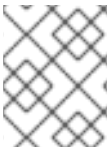
   Configure the following ports on both the front and back of the load balancers:

   Table 1.12. Application Ingress load balancer

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

TIP

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



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

NTP configuration

An OpenShift Container Platform cluster that is installed in a restricted network is configured to use a public Network Time Protocol (NTP) server by default. To avoid clock skew, reconfigure the cluster to use a private NTP server instead. For more information, see the documentation for *Configuring chrony time service*.

Additional resources

- Configuring chrony time service

1.2.4.2. User-provisioned DNS requirements

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

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

Table 1.13. Required DNS records

Component	Record	Description

Component	Record	Description
Kubernetes API	**api.<cluster_name>.<base_domain>.**	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster. **IMPORTANT** The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	***.apps.<cluster_name>.<base_domain>.**	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	**bootstrap.<cluster_name>.<base_domain>.**	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	**<master><n>.<cluster_name>.<base_domain>.**	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	**<worker><n>.<cluster_name>.<base_domain>.**	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 1.3. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

The following example BIND zone file shows sample PTR records for reverse name resolution.

Example 1.4. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
```

```
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

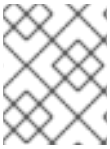
1.2.5. Generating an SSH private key and adding it to the agent

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

NOTE

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

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



NOTE

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

Procedure

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

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 Specify the path and file name, such as ~/**.ssh/id_rsa**, of the new SSH key.

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

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

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

   Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.2.6. Obtaining the installation program

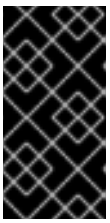
Before you install OpenShift Container Platform, download the installation file on a local computer.

Prerequisites

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

Procedure

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



IMPORTANT

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

IMPORTANT

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.

4. 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
```

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

1.2.7. Installing the OpenShift CLI by downloading the binary

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

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.6. Download and install the new version of **oc**.

1.2.7.1. Installing the OpenShift CLI on Linux

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

Procedure

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

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

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.2.7.2. Installing the OpenShift CLI on Windows

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

Procedure

1. Navigate to the Infrastructure Provider page on the Red Hat OpenShift Cluster Manager site.

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

```
C:\> path
```

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

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

1.2.7.3. Installing the OpenShift CLI on macOS

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

Procedure

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

```
$ echo $PATH
```

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

```
$ oc <command>
```

1.2.8. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

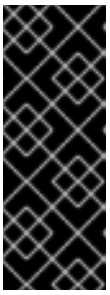
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.

Procedure

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

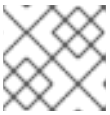
```
$ mkdir <installation_directory>
```



IMPORTANT

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

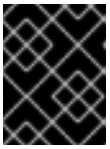
2. Customize the following **install-config.yaml** file template and save it in the **<installation_directory>**.



NOTE

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

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



IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

1.2.8.1. Sample install-config.yaml file for bare metal

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

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master (7)
  replicas: 3 (8)
metadata:
  name: test (9)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (10)
    hostPrefix: 23 (11)
  networkType: OpenShiftSDN
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
```

```
fips: false 14
pullSecret: '{"auths": ...}' 15
sshKey: 'ssh-ed25519 AAAA...' 16
```

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

2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

3 6 7 Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.

IMPORTANT

If you disable simultaneous multithreading (SMT) using the **install-config.yaml** file or in your BIOS settings, ensure that your capacity planning accounts for the dramatically decreased machine performance. If you disable SMT in the BIOS, the **hyperthreading** parameter in the **install-config.yaml** file will have no effect.

4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.

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

9 The cluster name that you specified in your DNS records.

10 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.

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

12 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.

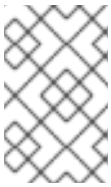
13 You must set the platform to **none**. You cannot provide additional platform configuration variables for your platform.

14 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container

Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.

15 The pull secret that you obtained from the Pull Secret page on the Red Hat OpenShift Cluster Manager site. This pull secret allows you to authenticate with the services that are provided by the included authorities, including Quay.io, which serves the container images for OpenShift Container Platform components.

16 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

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

1.2.8.2. Network configuration parameters

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

NOTE

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

Table 1.14. Required network parameters

Parameter	Description	Value
networking.networkType	The default Container Network Interface (CNI) network provider plug-in to deploy.	Either **OpenShiftSDN** or **OVNKubernetes**. The default value is **OpenShiftSDN**.
networking.clusterNetwork[].cidr	A block of IP addresses from which pod IP addresses are allocated. The **OpenShiftSDN** network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload.	An IP address allocation in CIDR format. The default value is **10.128.0.0/14**.
networking.clusterNetwork[].hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a /**23** subnet out of the given **cidr**, allowing for 510 (2^(32 - 23) - 2) pod IP addresses.	A subnet prefix. The default value is **23**.
networking.serviceNetwork[]	A block of IP addresses for services. **OpenShiftSDN** allows only one **serviceNetwork** block. The address block must not overlap with any other network block.	An IP address allocation in CIDR format. The default value is **172.30.0.0/16**.

Parameter	Description	Value
networking.machineNetwork[].cidr	A block of IP addresses assigned to nodes created by the OpenShift Container Platform installation program while installing the cluster. The address block must not overlap with any other network block. Multiple CIDR ranges may be specified.	An IP address allocation in CIDR format. The default value is **10.0.0.0/16**.

1.2.9. Modifying advanced network configuration parameters

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

IMPORTANT

Modifying the OpenShift Container Platform manifest files directly is not supported.

Prerequisites

- Create the **install-config.yaml** file and complete any modifications to it.
- Create the Ignition config files for your cluster.

Procedure

1. Change to the directory that contains the installation program and create the manifests:

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

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

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

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

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

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

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

Example output

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

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

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

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

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

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

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

1.2.10. Cluster Network Operator configuration

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

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

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
```

```
- cidr: 10.128.0.0/14
  hostPrefix: 23
serviceNetwork: 2
- 172.30.0.0/16
defaultNetwork: 3
  ...
kubeProxyConfig: 4
  iptablesSyncPeriod: 30s 5
  proxyArguments:
    iptables-min-sync-period: 6
    - 0s
```

1 2 Specified in the **install-config.yaml** file.

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

4 The parameters for this object specify the **kube-proxy** configuration. If you do not specify the parameter values, the Cluster Network Operator applies the displayed default parameter values. If you are using the OVN-Kubernetes default CNI network provider, the kube-proxy configuration has no effect.

5 The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the Go time package documentation.

NOTE

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

6 The minimum duration before refreshing **iptables** rules. This parameter ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the Go time package.

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

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

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

1 Specified in the **install-config.yaml** file.

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

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

4 The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

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

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

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

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

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

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

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

1 Specified in the **install-config.yaml** file.

2 Specify only if you want to override part of the OVN-Kubernetes configuration.

3 The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.

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

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

4 The UDP port for the Geneve overlay network.

1.2.10.3. Cluster Network Operator example configuration

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

Cluster Network Operator example CR

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

1.2.11. Creating the Ignition config files

Because you must manually start the cluster machines, you must generate the Ignition config files that the cluster needs to make its machines.

IMPORTANT

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

Prerequisites

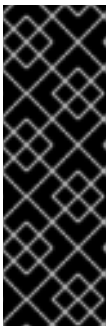
- Obtain the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

- Obtain the Ignition config files:

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

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



IMPORTANT

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

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

1.2.12. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. To create the machines, follow either the steps to use an ISO image or network PXE booting.

There are several methods of configuring RHCOS during ISO and PXE installations. These include:

- Kernel arguments: For a PXE install, you can **APPEND** arguments to the kernel of the live installer. For an ISO install, you can interrupt the live installation boot process to add kernel arguments. In both cases, you can use special **coreos.inst.*** arguments to direct the installer, as well as standard boot arguments for turning standard kernel services on or off.

- Ignition configs: You must generate an OpenShift Container Platform Ignition config file (***.ign**) for the type of node you are installing (worker, control plane, or bootstrap). You pass the location of the Ignition config to the installed system 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 installer and should not be used to include the standard **worker** and **master** Ignition configs.

- **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.

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.

1.2.12.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

Before you install a cluster on infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Have access to an HTTP server that can be accessed from your computer, and from the machines that you create.

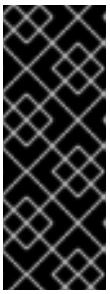
Procedure

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

IMPORTANT

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

2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the RHCOS image mirror page.



IMPORTANT

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

3. Use the ISO to start the RHCOS installation. Use one of the following installation options:
   - Burn the ISO image to a disk and boot it directly.
   - Use ISO redirection via a LOM interface.
4. Boot the ISO image. You can interrupt the installation boot process to add kernel arguments. However, for this ISO procedure you should use the **coreos-installer** command instead of adding kernel arguments. If you run the live installer without options or interruption, the installer boots up to a shell prompt on the live system, ready for you to install RHCOS to disk.
5. Review the *Advanced RHCOS installation reference* section for different ways of configuring features, such as networking and disk partitions, before running the **coreos-installer**.
6. Run the **coreos-installer** command. At a minimum, you must identify the Ignition config file location for your node type, and the location of the disk you are installing to. Here is an example:

```
$ sudo coreos-installer install \
    --ignition-url=https://host/worker.ign /dev/sda
```

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

8. 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, which is the default, also create at least two compute machines before you install the cluster.

1.2.12.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

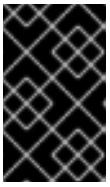
Before you install a cluster that uses manually-provisioned RHCOS nodes, such as bare metal, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.
- Configure suitable PXE or iPXE infrastructure.
- Have access to an HTTP server that you can access from your computer.

Procedure

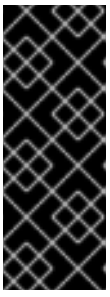
1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

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. Obtain the RHCOS **kernel**, **initramfs** and **rootfs** files from the Product Downloads page on the Red Hat customer portal or the RHCOS image mirror page.



IMPORTANT

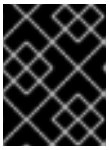
The RHCOS artifacts might not change with every release of OpenShift Container Platform. You must download artifacts with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Only use the appropriate **kernel**, **initramfs**, and **rootfs** artifacts 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**

3. 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.

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

5. Configure PXE or iPXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> 1
    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 2 3
```

1. 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.

2. If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

3. Specify locations of the RHCOS files that you uploaded to your HTTP 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?.

- For iPXE:

```
kernel  http://<HTTP_server>/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 1 2
initrd http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
boot
```

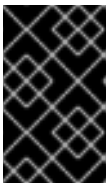
1. Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value is the location of the **initramfs** file. The **coreos.live.rootfs_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.

2. If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

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?.

6. Continue to create the machines for your cluster.



IMPORTANT

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

1.2.12.3. Advanced Red Hat Enterprise Linux CoreOS (RHCOS) installation configuration

A key benefit for manually provisioning the Red Hat Enterprise Linux CoreOS (RHCOS) nodes for OpenShift Container Platform is to be able to do configuration that is not available through default OpenShift Container Platform installation methods. This section describes some of the configurations that you can do using techniques that include:

- Passing kernel arguments to the live installer
- Running **coreos-installer** manually from the live system
- Embedding Ignition configs in an ISO

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

1.2.12.3.1. Using advanced networking options for PXE and ISO installations

Networking for OpenShift Container Platform nodes uses DHCP by default to gather all necessary configuration settings. To set up static IP addresses or configure special settings, such as bonding, you can do one of the following:

- Pass special kernel parameters when you boot the live installer.
- Use a machine config to copy networking files to the installed system.
- Configure networking from a live installer shell prompt, then copy those settings to the installed system so that they take effect when the installed system first boots.

To configure a PXE or iPXE installation, use one of the following options:

- See the "Advanced RHCOS installation reference" tables.
- Use a machine config to copy networking files to the installed system.

To configure an ISO installation, use the following procedure.

Procedure

1. Boot the ISO installer.
2. From the live system shell prompt, configure networking for the live system using available RHEL tools, such as **nmcli** or **nmtui**.
3. Run the **coreos-installer** command to install the system, adding the **--copy-network** option to copy networking configuration. For example:

```
$ coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```

4. Reboot into the installed system.

1.2.12.3.2. Disk partitioning

In most cases, data partitions are originally created by installing RHCOS, rather than by installing another operating system. In such cases, the OpenShift Container Platform installer should be allowed to configure your disk partitions.

However, there are two cases where you might want to intervene to override the default partitioning when installing an OpenShift Container Platform node:

- Create separate partitions: For greenfield installations on an empty disk, you might want to add separate storage to a partition. This is officially supported for making **/var** or a subdirectory of **/var**, such as **/var/lib/etcd**, a separate partition, but not both.

IMPORTANT

Kubernetes supports only two filesystem partitions. If you add more than one partition to the original configuration, Kubernetes cannot monitor all of them.

- Retain existing partitions: For a brownfield installation where you are reinstalling OpenShift Container Platform on an existing node and want to retain data partitions installed from your previous operating system, there are both boot arguments and options to **coreos-installer** that

allow you to retain existing data partitions.

1.2.12.3.2.1. Creating a separate /var partition

In general, disk partitioning for OpenShift Container Platform should be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.
- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.
- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a **/var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because **/var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate **/var** partition by creating a machine config that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. Create a **MachineConfig** object and add it to a file in the **openshift** directory. For example, name the file **98-var-partition.yaml**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This attaches storage to a separate **/var** directory.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
```

```
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      disks:
      - device: /dev/<device_name> 1
        partitions:
        - sizeMiB: <partition_size>
          startMiB: <partition_start_offset> 2
          label: var
      filesystems:
        - path: /var
          device: /dev/disk/by-partlabel/var
          format: xfs
    systemd:
      units:
        - name: var.mount
          enabled: true
          contents: |
            [Unit]
            Before=local-fs.target
            [Mount]
            Where=/var
            What=/dev/disk/by-partlabel/var
            [Install]
            WantedBy=local-fs.target
```

1 The storage device name of the disk that you want to partition.

2 When adding a data partition to the boot disk, a minimum value of 25000 mebibytes is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.

4. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Now you can use the Ignition config files as input to the ISO or PXE manual installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

1.2.12.3.2.2. Retaining existing partitions

For an ISO installation, you can add options to the **coreos-installer** command line that causes the installer to maintain one or more existing partitions. For a PXE installation, you can **APPEND coreos.inst.*** options to preserve partitions.

Saved partitions might be partitions from an existing OpenShift Container Platform system that has data partitions that you want to keep. Here are a few tips:

- If you save existing partitions, and those partitions do not leave enough space for RHCOS, installation will fail without damaging the saved partitions.
- Identify the disk partitions you want to keep either by partition label or by number.

For an ISO installation

This example preserves any partition in which the partition label begins with **data** (**data***):

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

The following example illustrates running the **coreos-installer** in a way that preserves the sixth (6) partition on the disk:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

This example preserves partitions 5 and higher:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

In the previous examples where partition saving is used, **coreos-installer** recreates the partition immediately.

For a PXE installation

This **APPEND** option preserves any partition in which the partition label begins with 'data' ('data*'):

```
coreos.inst.save_partlabel=data*
```

This **APPEND** option preserves partitions 5 and higher:

```
coreos.inst.save_partindex=5-
```

This **APPEND** option preserves partition 6:

```
coreos.inst.save_partindex=6
```

1.2.12.3.3. Identifying Ignition configs

When doing an RHCOS manual installation, there are two types of Ignition configs that you can provide, with different reasons for providing each one:

- **Permanent install Ignition config**: Every manual RHCOS installation needs to pass one of the Ignition config files generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, to carry out the installation.

IMPORTANT

It is not recommended to modify these files.

For PXE installations, you pass the Ignition configs on the **APPEND** line using the **coreos.inst.ignition_url=** option. For ISO installations, after the ISO boots to the shell prompt, you identify the Ignition config on the **coreos-installer** command line with the **--ignition-url=** option. In both cases, only HTTP and HTTPS protocols are supported.

- **Live install Ignition config**: This type must be created manually and should be avoided if possible, as it is not supported by Red Hat. With this method, the Ignition config passes to the live install medium, runs immediately upon booting, and performs setup tasks before and/or after the RHCOS system installs to disk. This method should only be used for performing tasks that must be performed once and not applied again later, such as with advanced partitioning that cannot be done using a machine config.
  For PXE or ISO boots, you can create the Ignition config and **APPEND** the **ignition.config.url=** option to identify the location of the Ignition config. You also need to append **ignition.firstboot ignition.platform.id=metal** or the **ignition.config.url** option will be ignored.

1.2.12.3.3.1. Embedding an Ignition config in the RHCOS ISO

You can embed a live install Ignition config directly in an RHCOS ISO image. When the ISO image is booted, the embedded config will be applied automatically.

Procedure

1. Download the **coreos-installer** binary from the following image mirror page: https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.
2. Retrieve the RHCOS ISO image and the Ignition config file, and copy them into an accessible directory, such as **/mnt**:

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. Run the following command to embed the Ignition config into the ISO:

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

You can now use that ISO to install RHCOS using the specified live install Ignition config.

IMPORTANT

Using **coreos-installer iso ignition embed** to embed a file generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, is unsupported and not recommended.

4. To show the contents of the embedded Ignition config and direct it into a file, run:

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

Example output

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. To remove the Ignition config and return the ISO to its pristine state so you can reuse it, run:

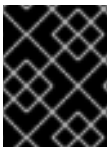
```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

You can now embed another Ignition config into the ISO or use the ISO in its pristine state.

1.2.12.3.4. Advanced RHCOS installation reference

This section illustrates the networking configuration and other advanced options that allow you to modify the Red Hat Enterprise Linux CoreOS (RHCOS) manual installation process. The following tables describe the kernel arguments and command-line options you can use with the RHCOS live installer and the **coreos-installer** command.

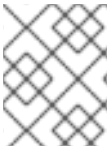
Routing and bonding options at RHCOS boot prompt
If you install RHCOS from an ISO image, you can add kernel arguments manually when you boot that image to configure the node's networking. If no networking arguments are used, the installation defaults to using DHCP.



IMPORTANT

When adding networking arguments, you must also add the **rd.neednet=1** kernel argument.

The following table describes how to use **ip=**, **nameserver=**, and **bond=** kernel arguments for live ISO installs.



NOTE

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

Table 1.15. Routing and bonding options for ISO

Description	Examples
To configure an IP address, either use DHCP (**ip=dhcp**) or set an individual static IP address (**ip=<host_ip>**). Then identify the DNS server IP address (**nameserver=<dns_ip>**) on each node. This example sets: • 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**	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41
Specify multiple network interfaces by specifying multiple **ip=** entries.	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
Disable DHCP on a single interface, such as when there are two or more network interfaces and only one interface is being 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
You can combine DHCP and static IP configurations on systems with multiple network interfaces.	ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
You can provide multiple DNS servers by adding a **nameserver=** entry for each server.	nameserver=1.1.1.1 nameserver=8.8.8.8

Description	Examples

Optional: Bonding multiple network interfaces to a single interface is supported using the **bond=** option. In these two examples:
• The syntax for configuring a bonded interface is: **bond=name[:network_interfaces][:options]**
• *name* is the bonding device name (**bond0**), *network_interfaces* represents a comma-separated list of physical (ethernet) interfaces (**em1,em2**), 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:em1,em2: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:em1,em2:mode=active-backup`
	`ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none`

coreos.inst boot options for ISO or PXE install

While you can pass most standard installation boot arguments to the live installer, there are several arguments that are specific to the RHCOS live installer.

- For ISO, these options can be added by interrupting the RHCOS installer.
- For PXE or iPXE, these options must be added to the **APPEND** line before starting the PXE kernel. You cannot interrupt a live PXE install.

The following table shows the RHCOS live installer boot options for ISO and PXE installs.

Table 1.16. coreos.inst boot options

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to. 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.
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.

Argument	Description
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.
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. Once 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.

coreos-installer options for ISO install

You can also install RHCOS by invoking the **coreos-installer** command directly from the command line. The kernel arguments in the previous table provide a shortcut for automatically invoking **coreos-installer** at boot time, but you can pass similar arguments directly to **coreos-installer** when running it from a shell prompt.

The following table shows the options and subcommands you can pass to the **coreos-installer** command from a shell prompt during a live install.

Table 1.17. coreos-installer command-line options, arguments, and subcommands

Command-line options	
Option	**Description**
-u, **--image-url <url>**	Specify the image URL manually.
-f, **--image-file <path>**	Specify a local image file manually.
-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.
--append-karg <arg>…	Append the default kernel argument.
--delete-karg <arg>…	Delete the default kernel argument.
-n, **--copy-network**	Copy the network configuration from the install environment.
--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.
--offline	Force offline installation.
--insecure	Skip signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Default is **x86_64**.
--preserve-on-error	Do not clear partition table on error.

-h, **--help**	Print help information.
Command-line argument	
Argument	Description
<device>	The destination device.
coreos-installer embedded Ignition commands	
Command	Description
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	Embed an Ignition config in an ISO image.
coreos-installer iso ignition show <options> <ISO_image>	Show the embedded Ignition config from an ISO image.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.
coreos-installer ISO Ignition options	
Option	Description
-f, **--force**	Overwrite an existing Ignition config.
-i, **--ignition-file <path>**	The Ignition config to be used. Default is **stdin**.
-o, **--output <path>**	Write the ISO to a new output file.
-h, **--help**	Print help information.
coreos-installer PXE Ignition commands	
Command	Description
Note that not all of these options are accepted by all subcommands.	
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 ignition unwrap <options> <initrd_name>	Show the wrapped Ignition config in an **initrd** image.

coreos-installer PXE Ignition options	
Option	**Description**
-i, --ignition-file <path>	The Ignition config to be used. Default is **stdin**.
-o, --output <path>	Write the ISO to a new output file.
-h, --help	Print help information.

1.2.13. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.
- Your machines have direct Internet access or have an HTTP or HTTPS proxy available.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

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

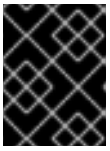
2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.19.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

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

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



IMPORTANT

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

1.2.14. Logging in to the cluster by using the CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.2.15. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.19.0
master-1  Ready     master  63m  v1.19.0
master-2  Ready     master  64m  v1.19.0
worker-0  NotReady  worker  76s  v1.19.0
worker-1  NotReady  worker  70s  v1.19.0
```

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:

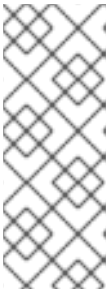
```
$ oc get csr
```

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**.

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> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

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

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

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> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

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

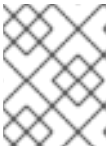
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

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

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



NOTE

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

Additional information

- For more information on CSRs, see Certificate Signing Requests.

1.2.16. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

```
NAME                                 VERSION AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                         4.6.0   True        False         False      3h56m
cloud-credential                       4.6.0   True        False         False      29h
cluster-autoscaler                     4.6.0   True        False         False      29h
config-operator                        4.6.0   True        False         False      6h39m
console                              4.6.0   True        False         False      3h59m
```

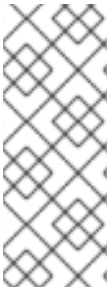
```
csi-snapshot-controller                 4.6.0   True        False         False      4h12m
dns                                     4.6.0   True        False         False      4h15m
etcd                                    4.6.0   True        False         False      29h
image-registry                          4.6.0   True        False         False      3h59m
ingress                                 4.6.0   True        False         False      4h30m
insights                                4.6.0   True        False         False      29h
kube-apiserver                          4.6.0   True        False         False      29h
kube-controller-manager                 4.6.0   True        False         False      29h
kube-scheduler                          4.6.0   True        False         False      29h
kube-storage-version-migrator           4.6.0   True        False         False      4h2m
machine-api                             4.6.0   True        False         False      29h
machine-approver                        4.6.0   True        False         False      6h34m
machine-config                          4.6.0   True        False         False      3h56m
marketplace                             4.6.0   True        False         False      4h2m
monitoring                              4.6.0   True        False         False      6h31m
network                                 4.6.0   True        False         False      29h
node-tuning                             4.6.0   True        False         False      4h30m
openshift-apiserver                     4.6.0   True        False         False      3h56m
openshift-controller-manager            4.6.0   True        False         False      4h36m
openshift-samples                       4.6.0   True        False         False      4h30m
operator-lifecycle-manager              4.6.0   True        False         False      29h
operator-lifecycle-manager-catalog      4.6.0   True        False         False      29h
operator-lifecycle-manager-packageserver 4.6.0  True        False         False      3h59m
service-ca                              4.6.0   True        False         False      29h
storage                                 4.6.0   True        False         False      4h30m
```

2. Configure the Operators that are not available.

1.2.16.1. Image registry removed during installation

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

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



NOTE

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing configs.imageregistry.operator.openshift.io."

1.2.16.2. Image registry storage configuration

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

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

1.2.16.3. Configuring block registry storage

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

IMPORTANT

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

Procedure

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

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

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

3. Edit the registry configuration so that it references the correct PVC.

1.2.17. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

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

Procedure

1. Confirm that all the cluster components are online:

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

Example output

```
NAME                                       VERSION AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.6.0   True        False         False      3h56m
cloud-credential                            4.6.0   True        False         False      29h
cluster-autoscaler                          4.6.0   True        False         False      29h
config-operator                             4.6.0   True        False         False      6h39m
console                                    4.6.0   True        False         False      3h59m
```

```
csi-snapshot-controller                 4.6.0   True        False         False      4h12m
dns                                     4.6.0   True        False         False      4h15m
etcd                                    4.6.0   True        False         False      29h
image-registry                          4.6.0   True        False         False      3h59m
ingress                                 4.6.0   True        False         False      4h30m
insights                                4.6.0   True        False         False      29h
kube-apiserver                          4.6.0   True        False         False      29h
kube-controller-manager                 4.6.0   True        False         False      29h
kube-scheduler                          4.6.0   True        False         False      29h
kube-storage-version-migrator           4.6.0   True        False         False      4h2m
machine-api                             4.6.0   True        False         False      29h
machine-approver                        4.6.0   True        False         False      6h34m
machine-config                          4.6.0   True        False         False      3h56m
marketplace                             4.6.0   True        False         False      4h2m
monitoring                              4.6.0   True        False         False      6h31m
network                                 4.6.0   True        False         False      29h
node-tuning                             4.6.0   True        False         False      4h30m
openshift-apiserver                     4.6.0   True        False         False      3h56m
openshift-controller-manager            4.6.0   True        False         False      4h36m
openshift-samples                       4.6.0   True        False         False      4h30m
operator-lifecycle-manager              4.6.0   True        False         False      29h
operator-lifecycle-manager-catalog      4.6.0   True        False         False      29h
operator-lifecycle-manager-packageserver 4.6.0  True        False         False      3h59m
service-ca                              4.6.0   True        False         False      29h
storage                                 4.6.0   True        False         False      4h30m
```

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

2. Monitor for cluster completion:

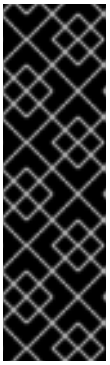
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

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

Example output

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

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



IMPORTANT

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

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

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

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

   Example output

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

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

   ```
   $ oc logs <pod_name> -n <namespace> 1
   ```

   1 Specify the pod name and namespace, as shown in the output of the previous command.

   If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

1.2.18. Next steps

- Customize your cluster.
- If necessary, you can opt out of remote health reporting .
- Set up your registry and configure registry storage .

1.3. INSTALLING A CLUSTER ON BARE METAL IN A RESTRICTED NETWORK

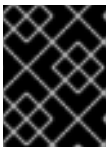
In OpenShift Container Platform version 4.6, you can install a cluster on bare metal infrastructure that you provision in a restricted network.

IMPORTANT

While you might be able to follow this procedure to deploy a cluster on virtualized or cloud environments, you must be aware of additional considerations for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you attempt to install an OpenShift Container Platform cluster in such an environment.

1.3.1. Prerequisites

- Create a registry on your mirror host and obtain the **imageContentSources** data for your version of OpenShift Container Platform.



IMPORTANT

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

- Provision persistent storage for your cluster. To deploy a private image registry, your storage must provide ReadWriteMany access modes.
- Review details about the OpenShift Container Platform installation and update processes.
- If you use a firewall and plan to use telemetry, you must configure the firewall to allow the sites that your cluster requires access to.

NOTE

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

1.3.2. About installations in restricted networks

In OpenShift Container Platform 4.6, you can perform an installation that does not require an active connection to the Internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

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

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

IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

1.3.2.1. Additional limits

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

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

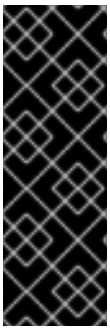
1.3.3. Internet and Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.6, you require access to the Internet to obtain the images that are necessary to install your cluster. The Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, also requires Internet access. If your cluster is connected to the Internet, Telemetry runs automatically, and your cluster is registered to the Red Hat OpenShift Cluster Manager (OCM).

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

You must have Internet access to:

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



IMPORTANT

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

1.3.4. Machine requirements for a cluster with user-provisioned infrastructure

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

1.3.4.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

- One temporary bootstrap machine
- Three control plane, or master, machines
- At least two compute machines, which are also known as worker machines. If you are running a three-node cluster, running zero compute machines is supported. Running one compute machine is not supported.

NOTE

The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster.

IMPORTANT

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

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 8 and inherits all of its hardware certifications and requirements. See Red Hat Enterprise Linux technology capabilities and limits .

1.3.4.2. Network connectivity requirements

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config files from the Machine Config Server. During the initial boot, the machines require either a DHCP server or that static IP addresses be set in order to establish a network connection to download their Ignition config files.

1.3.4.3. Minimum resource requirements

Each cluster machine must meet the following minimum requirements:

Machine	Operating System	vCPU[1]	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.8 - 7.9	2	8 GB	120 GB

[1] 1 physical core provides 2 vCPUs when hyper-threading is enabled. 1 physical core provides 1 vCPU when hyper-threading is not enabled.

1.3.4.4. Certificate signing requests management

Because your cluster has limited access to automatic machine management when you use infrastructure that you provision, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation. The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

1.3.5. Creating the user-provisioned infrastructure

Before you deploy an OpenShift Container Platform cluster that uses user-provisioned infrastructure, you must create the underlying infrastructure.

Prerequisites

- Review the OpenShift Container Platform 4.x Tested Integrations page before you create the supporting infrastructure for your cluster.

Procedure

1. Configure DHCP or set static IP addresses on each node.
2. Provision the required load balancers.
3. Configure the ports for your machines.
4. Configure DNS.
5. Ensure network connectivity.

1.3.5.1. Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in **initramfs** during boot to fetch Ignition config from the machine config server.

During the initial boot, the machines require either a DHCP server or that static IP addresses be set on each host in the cluster in order to establish a network connection, which allows them to download their Ignition config files.

It is recommended to use the DHCP server to manage the machines for the cluster long-term. Ensure that the DHCP server is configured to provide persistent IP addresses and host names to the cluster machines.

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.

You must configure the network connectivity between machines to allow cluster components to communicate. Each machine must be able to resolve the host names of all other machines in the cluster.

Table 1.18. All machines to all machines

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	**1936**	Metrics
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101** and the Cluster Version Operator on port**9099**.
	10250-**10259**	The default ports that Kubernetes reserves

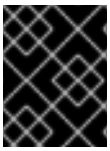
Protocol	Port	Description
	10256	openshift-sdn
UDP	**4789**	VXLAN and Geneve
	6081	VXLAN and Geneve
	9000-**9999**	Host level services, including the node exporter on ports **9100**-**9101**.
TCP/UDP	**30000**-**32767**	Kubernetes node port

Table 1.19. All machines to control plane

Protocol	Port	Description
TCP	**2379**-**2380**	etcd server, peer, and metrics ports
	6443	Kubernetes API

Network topology requirements

The infrastructure that you provision for your cluster must meet the following network topology requirements.



IMPORTANT

OpenShift Container Platform requires all nodes to have internet access to pull images for platform containers and provide telemetry data to Red Hat.

Load balancers

Before you install OpenShift Container Platform, you must provision two load balancers that meet the following requirements:

1. **API load balancer**: Provides a common endpoint for users, both human and machine, to interact with and configure the platform. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

   #### NOTE

   Session persistence is not required for the API load balancer to function properly.

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

Table 1.20. API load balancer

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

NOTE

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

2. **Application Ingress load balancer**: Provides an Ingress point for application traffic flowing in from outside the cluster. Configure the following conditions:

   - Layer 4 load balancing only. This can be referred to as Raw TCP, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the Ingress routes.
   - A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

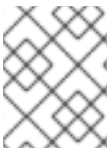
   Configure the following ports on both the front and back of the load balancers:

Table 1.21. Application Ingress load balancer

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

TIP

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



NOTE

A working configuration for the Ingress router is required for an OpenShift Container Platform cluster. You must configure the Ingress router after the control plane initializes.

NTP configuration

An OpenShift Container Platform cluster that is installed in a restricted network is configured to use a public Network Time Protocol (NTP) server by default. To avoid clock skew, reconfigure the cluster to use a private NTP server instead. For more information, see the documentation for *Configuring chrony time service*.

Additional resources

- Configuring chrony time service

1.3.5.2. User-provisioned DNS requirements

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

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

Table 1.22. Required DNS records

Component	Record	Description
Kubernetes API	**api.<cluster_name>.<base_domain>.**	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

Component	Record	Description
	api-int.<cluster_name>. <base_domain>.	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the load balancer for the control plane machines. These records must be resolvable from all the nodes within the cluster. **IMPORTANT** The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If the API server cannot resolve the node names, then proxied API calls can fail, and you cannot retrieve logs from pods.
Routes	***.apps.<cluster_name>. <base_domain>.**	Add a wildcard DNS A/AAAA or CNAME record that refers to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
Bootstrap	**bootstrap.<cluster_name>. <base_domain>.**	Add a DNS A/AAAA or CNAME record, and a DNS PTR record, to identify the bootstrap machine. These records must be resolvable by the nodes within the cluster.
Master hosts	**<master><n>. <cluster_name>. <base_domain>.**	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the master nodes. These records must be resolvable by the nodes within the cluster.
Worker hosts	**<worker><n>. <cluster_name>. <base_domain>.**	Add DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

TIP

You can use the **nslookup <hostname>** command to verify name resolution. You can use the **dig -x <ip_address>** command to verify reverse name resolution for the PTR records.

The following example of a BIND zone file shows sample A records for name resolution. The purpose of the example is to show the records that are needed. The example is not meant to provide advice for choosing one name resolution service over another.

Example 1.5. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

The following example BIND zone file shows sample PTR records for reverse name resolution.

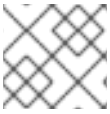
Example 1.6. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
```

```
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

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

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



NOTE

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

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

NOTE

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

Procedure

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

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 Specify the path and file name, such as ~/**.ssh/id_rsa**, of the new SSH key.

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

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

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

   Example output

```
Agent pid 31874
```

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

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

Example output

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

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

Next steps

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

1.3.7. Manually creating the installation configuration file

For installations of OpenShift Container Platform that use user-provisioned infrastructure, you manually generate your installation configuration file.

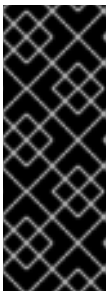
Prerequisites

- Obtain the OpenShift Container Platform installation program and the access token for your cluster.
- Obtain the **imageContentSources** section from the output of the command to mirror the repository.
- Obtain the contents of the certificate for your mirror registry.

Procedure

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

```
$ mkdir <installation_directory>
```



IMPORTANT

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

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

NOTE

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

- Unless you use a registry that RHCOS trusts by default, such as **docker.io**, you must provide the contents of the certificate for your mirror repository in the **additionalTrustBundle** section. In most cases, you must provide the certificate for your mirror.
- You must include the **imageContentSources** section from the output of the command to mirror the repository.

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

IMPORTANT

The **install-config.yaml** file is consumed during the next step of the installation process. You must back it up now.

1.3.7.1. Sample install-config.yaml file for bare metal

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

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master (7)
  replicas: 3 (8)
metadata:
  name: test (9)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (10)
    hostPrefix: 23 (11)
  networkType: OpenShiftSDN
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
additionalTrustBundle: | (17)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
```

```
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

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

2 5 The **controlPlane** section is a single mapping, but the compute section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Although both sections currently define a single machine pool, it is possible that future versions of OpenShift Container Platform will support defining multiple compute pools during installation. Only one control plane pool is used.

3 6 7 Whether to enable or disable simultaneous multithreading (SMT), or **hyperthreading**. By default, SMT is enabled to increase the performance of your machines' cores. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster machines; this includes both control plane and compute machines.

IMPORTANT

If you disable simultaneous multithreading (SMT) using the **install-config.yaml** file or in your BIOS settings, ensure that your capacity planning accounts for the dramatically decreased machine performance. If you disable SMT in the BIOS, the **hyperthreading** parameter in the **install-config.yaml** file will have no effect.

4 You must set the value of the **replicas** parameter to **0**. This parameter controls the number of workers that the cluster creates and manages for you, which are functions that the cluster does not perform when you use user-provisioned infrastructure. You must manually deploy worker machines for the cluster to use before you finish installing OpenShift Container Platform.

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

9 The cluster name that you specified in your DNS records.

10 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.

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

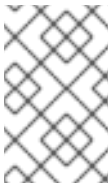
12 The IP address pool to use for service IP addresses. You can enter only one IP address pool. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.

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

14 Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container

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

16 The public portion of the default SSH key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).



NOTE

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

17 Provide the contents of the certificate file that you used for your mirror registry.

18 Provide the **imageContentSources** section from the output of the command to mirror the repository.

1.3.7.2. Configuring the cluster-wide proxy during installation

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

NOTE

For bare metal installations, if you do not assign node IP addresses from the range that is specified in the **networking.machineNetwork[].cidr** field in the **install-config.yaml** file, you must include them in the **proxy.noProxy** field.

Prerequisites

- You have an existing **install-config.yaml** file.
- You 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 Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

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

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

1. A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpProxy** value.

2. A proxy URL to use for creating HTTPS connections outside the cluster. If this field is not specified, then **httpProxy** is used for both HTTP and HTTPS connections. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must not specify an **httpsProxy** value.

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

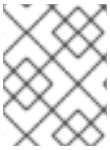
4. If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **Proxy** object's **trustedCA** field. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle. If you use an MITM transparent proxy network that does not require additional proxy configuration but requires additional CAs, you must provide the MITM CA certificate.

NOTE

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

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

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



NOTE

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

1.3.8. Configuring a three-node cluster

You can optionally install and run three-node clusters in OpenShift Container Platform with no workers. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for development, production, and testing.

Procedure

- Edit the **install-config.yaml** file to set the number of compute replicas, which are also known as worker replicas, to **0**, as shown in the following **compute** stanza:

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

1.3.9. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to make its machines.

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

IMPORTANT

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

Prerequisites

- You obtained the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- You created the **install-config.yaml** installation configuration file.

Procedure

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

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

Example output

```
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: install_dir/manifests and install_dir/openshift
```

1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

WARNING

If you are running a three-node cluster, skip the following step to allow the masters to be schedulable.

1. 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.
2. 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> 1
```

1 For **<installation_directory>**, specify the same installation directory.

The following files are generated in the directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

1.3.10. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines

Before you install a cluster on bare metal infrastructure that you provision, you must create RHCOS machines for it to use. To create the machines, follow either the steps to use an ISO image or network PXE booting.

There are several methods of configuring RHCOS during ISO and PXE installations. These include:

- Kernel arguments: For a PXE install, you can **APPEND** arguments to the kernel of the live installer. For an ISO install, you can interrupt the live installation boot process to add kernel arguments. In both 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: You must generate an OpenShift Container Platform Ignition config file (***.ign**) for the type of node you are installing (worker, control plane, or bootstrap). You pass the location of the Ignition config to the installed system 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 installer and should not be used to include the standard **worker** and **master** Ignition configs.

- **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.

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.

1.3.10.1. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines using an ISO image

Before you install a cluster on infrastructure that you provision, you must create RHCOS machines for it to use. You can use an ISO image to create the machines.

Prerequisites

- Obtain the Ignition config files for your cluster.

- Have access to an HTTP server that can be accessed from your computer, and from the machines that you create.

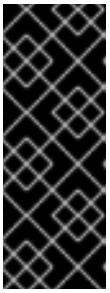
Procedure

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

IMPORTANT

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

2. Obtain the RHCOS images that are required for your preferred method of installing operating system instances from the RHCOS image mirror page.



IMPORTANT

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

3. Use the ISO to start the RHCOS installation. Use one of the following installation options:
   - Burn the ISO image to a disk and boot it directly.
   - Use ISO redirection via a LOM interface.
4. Boot the ISO image. You can interrupt the installation boot process to add kernel arguments. However, for this ISO procedure you should use the **coreos-installer** command instead of adding kernel arguments. If you run the live installer without options or interruption, the installer boots up to a shell prompt on the live system, ready for you to install RHCOS to disk.
5. Review the *Advanced RHCOS installation reference* section for different ways of configuring features, such as networking and disk partitions, before running the **coreos-installer**.
6. Run the **coreos-installer** command. At a minimum, you must identify the Ignition config file location for your node type, and the location of the disk you are installing to. Here is an example:

```
$ sudo coreos-installer install \
    --ignition-url=https://host/worker.ign /dev/sda
```

7. After RHCOS installs, the system reboots. During the system reboot, it applies the Ignition config file that you specified.
8. 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, which is the default, also create at least two compute machines before you install the cluster.

1.3.10.2. Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines by PXE or iPXE booting

Before you install a cluster that uses manually-provisioned RHCOS nodes, such as bare metal, you must create RHCOS machines for it to use. You can use PXE or iPXE booting to create the machines.

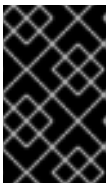
Prerequisites

- Obtain the Ignition config files for your cluster.
- Configure suitable PXE or iPXE infrastructure.

- Have access to an HTTP server that you can access from your computer.

Procedure

1. Upload the master, worker, and bootstrap Ignition config files that the installation program created to your HTTP server. Note the URLs of these files.



IMPORTANT

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. Obtain the RHCOS **kernel**, **initramfs** and **rootfs** files from the Product Downloads page on the Red Hat customer portal or the RHCOS image mirror page.

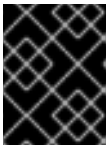
IMPORTANT

The RHCOS artifacts might not change with every release of OpenShift Container Platform. You must download artifacts 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**

3. 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.

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

5. Configure PXE or iPXE installation for the RHCOS images.
Modify one of the following example menu entries for your environment and verify that the image and Ignition files are properly accessible:

- For PXE:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
```

```
    KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
    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 (2) (3)
```

1. 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.

2. If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

3. Specify locations of the RHCOS files that you uploaded to your HTTP 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?.

- For iPXE:

```
kernel  http://<HTTP_server>/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 (1) (2)
initrd http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img
boot
```

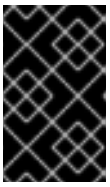
1. Specify locations of the RHCOS files that you uploaded to your HTTP server. The **kernel** parameter value is the location of the **kernel** file, the **initrd** parameter value is the location of the **initramfs** file. The **coreos.live.rootfs_url** parameter value is the location of the **rootfs** file, and the **coreos.inst.ignition_url** parameter value is the location of the bootstrap Ignition config file.

2. If you use multiple NICs, specify a single interface in the **ip** option. For example, to use DHCP on a NIC that is named **eno1**, set **ip=eno1:dhcp**.

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?.

6. Continue to create the machines for your cluster.



IMPORTANT

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

1.3.10.3. Advanced Red Hat Enterprise Linux CoreOS (RHCOS) installation configuration

A key benefit for manually provisioning the Red Hat Enterprise Linux CoreOS (RHCOS) nodes for OpenShift Container Platform is to be able to do configuration that is not available through default OpenShift Container Platform installation methods. This section describes some of the configurations that you can do using techniques that include:

- Passing kernel arguments to the live installer
- Running **coreos-installer** manually from the live system
- Embedding Ignition configs in an ISO

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

1.3.10.3.1. Using advanced networking options for PXE and ISO installations

Networking for OpenShift Container Platform nodes uses DHCP by default to gather all necessary configuration settings. To set up static IP addresses or configure special settings, such as bonding, you can do one of the following:

- Pass special kernel parameters when you boot the live installer.
- Use a machine config to copy networking files to the installed system.
- Configure networking from a live installer shell prompt, then copy those settings to the installed system so that they take effect when the installed system first boots.

To configure a PXE or iPXE installation, use one of the following options:

- See the "Advanced RHCOS installation reference" tables.
- Use a machine config to copy networking files to the installed system.

To configure an ISO installation, use the following procedure.

Procedure

1. Boot the ISO installer.
2. From the live system shell prompt, configure networking for the live system using available RHEL tools, such as **nmcli** or **nmtui**.
3. Run the **coreos-installer** command to install the system, adding the **--copy-network** option to copy networking configuration. For example:

```
$ coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```

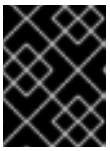
4. Reboot into the installed system.

1.3.10.3.2. Disk partitioning

In most cases, data partitions are originally created by installing RHCOS, rather than by installing another operating system. In such cases, the OpenShift Container Platform installer should be allowed to configure your disk partitions.

However, there are two cases where you might want to intervene to override the default partitioning when installing an OpenShift Container Platform node:

- Create separate partitions: For greenfield installations on an empty disk, you might want to add separate storage to a partition. This is officially supported for making **/var** or a subdirectory of **/var**, such as **/var/lib/etcd**, a separate partition, but not both.



IMPORTANT

Kubernetes supports only two filesystem partitions. If you add more than one partition to the original configuration, Kubernetes cannot monitor all of them.

- Retain existing partitions: For a brownfield installation where you are reinstalling OpenShift Container Platform on an existing node and want to retain data partitions installed from your previous operating system, there are both boot arguments and options to **coreos-installer** that allow you to retain existing data partitions.

1.3.10.3.2.1. Creating a separate /var partition

In general, disk partitioning for OpenShift Container Platform should be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.
- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.
- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a /**var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because /**var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate /**var** partition by creating a machine config that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
? SSH Public Key ...
$ ls $HOME/clusterconfig/openshift/
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

3. Create a **MachineConfig** object and add it to a file in the **openshift** directory. For example, name the file **98-var-partition.yaml**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This attaches storage to a separate /**var** directory.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      disks:
      - device: /dev/<device_name> 1
        partitions:
        - sizeMiB: <partition_size>
          startMiB: <partition_start_offset> 2
          label: var
      filesystems:
        - path: /var
          device: /dev/disk/by-partlabel/var
          format: xfs
    systemd:
      units:
```

```
    - name: var.mount
      enabled: true
      contents: |
        [Unit]
        Before=local-fs.target
        [Mount]
        Where=/var
        What=/dev/disk/by-partlabel/var
        [Install]
        WantedBy=local-fs.target
```

1. The storage device name of the disk that you want to partition.
2. When adding a data partition to the boot disk, a minimum value of 25000 mebibytes is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.

4. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

Now you can use the Ignition config files as input to the ISO or PXE manual installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

1.3.10.3.2.2. Retaining existing partitions

For an ISO installation, you can add options to the **coreos-installer** command line that causes the installer to maintain one or more existing partitions. For a PXE installation, you can **APPEND coreos.inst.*** options to preserve partitions.

Saved partitions might be partitions from an existing OpenShift Container Platform system that has data partitions that you want to keep. Here are a few tips:

- If you save existing partitions, and those partitions do not leave enough space for RHCOS, installation will fail without damaging the saved partitions.
- Identify the disk partitions you want to keep either by partition label or by number.

For an ISO installation

This example preserves any partition in which the partition label begins with **data** (**data***):

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partlabel 'data*' /dev/sda
```

The following example illustrates running the **coreos-installer** in a way that preserves the sixth (6) partition on the disk:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

This example preserves partitions 5 and higher:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

In the previous examples where partition saving is used, **coreos-installer** recreates the partition immediately.

For a PXE installation

This **APPEND** option preserves any partition in which the partition label begins with 'data' ('data*'):

```
coreos.inst.save_partlabel=data*
```

This **APPEND** option preserves partitions 5 and higher:

```
coreos.inst.save_partindex=5-
```

This **APPEND** option preserves partition 6:

```
coreos.inst.save_partindex=6
```

1.3.10.3.3. Identifying Ignition configs

When doing an RHCOS manual installation, there are two types of Ignition configs that you can provide, with different reasons for providing each one:

- **Permanent install Ignition config**: Every manual RHCOS installation needs to pass one of the Ignition config files generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, to carry out the installation.

  IMPORTANT

  It is not recommended to modify these files.

  For PXE installations, you pass the Ignition configs on the **APPEND** line using the **coreos.inst.ignition_url=** option. For ISO installations, after the ISO boots to the shell prompt, you identify the Ignition config on the **coreos-installer** command line with the **--ignition-url=** option. In both cases, only HTTP and HTTPS protocols are supported.

- **Live install Ignition config**: This type must be created manually and should be avoided if possible, as it is not supported by Red Hat. With this method, the Ignition config passes to the live install medium, runs immediately upon booting, and performs setup tasks before and/or after the RHCOS system installs to disk. This method should only be used for performing tasks that must be performed once and not applied again later, such as with advanced partitioning that cannot be done using a machine config.
  For PXE or ISO boots, you can create the Ignition config and **APPEND** the **ignition.config.url=** option to identify the location of the Ignition config. You also need to append **ignition.firstboot ignition.platform.id=metal** or the **ignition.config.url** option will be ignored.

1.3.10.3.3.1. Embedding an Ignition config in the RHCOS ISO

You can embed a live install Ignition config directly in an RHCOS ISO image. When the ISO image is booted, the embedded config will be applied automatically.

Procedure

1. Download the **coreos-installer** binary from the following image mirror page: https://mirror.openshift.com/pub/openshift-v4/clients/coreos-installer/latest/.

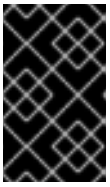
2. Retrieve the RHCOS ISO image and the Ignition config file, and copy them into an accessible directory, such as **/mnt**:

```
# cp rhcos-<version>-live.x86_64.iso bootstrap.ign /mnt/
# chmod 644 /mnt/rhcos-<version>-live.x86_64.iso
```

3. Run the following command to embed the Ignition config into the ISO:

```
# ./coreos-installer iso ignition embed -i /mnt/bootstrap.ign \
    /mnt/rhcos-<version>-live.x86_64.iso
```

   You can now use that ISO to install RHCOS using the specified live install Ignition config.



IMPORTANT

Using **coreos-installer iso ignition embed** to embed a file generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, is unsupported and not recommended.

4. To show the contents of the embedded Ignition config and direct it into a file, run:

```
# ./coreos-installer iso ignition show /mnt/rhcos-<version>-live.x86_64.iso > mybootstrap.ign
```

```
# diff -s bootstrap.ign mybootstrap.ign
```

Example output

```
Files bootstrap.ign and mybootstrap.ign are identical
```

5. To remove the Ignition config and return the ISO to its pristine state so you can reuse it, run:

```
# ./coreos-installer iso ignition remove /mnt/rhcos-<version>-live.x86_64.iso
```

   You can now embed another Ignition config into the ISO or use the ISO in its pristine state.

1.3.10.3.4. Advanced RHCOS installation reference

This section illustrates the networking configuration and other advanced options that allow you to modify the Red Hat Enterprise Linux CoreOS (RHCOS) manual installation process. The following tables describe the kernel arguments and command-line options you can use with the RHCOS live installer and the **coreos-installer** command.

Routing and bonding options at RHCOS boot prompt

If you install RHCOS from an ISO image, you can add kernel arguments manually when you boot that image to configure the node's networking. If no networking arguments are used, the installation defaults to using DHCP.

IMPORTANT

When adding networking arguments, you must also add the **rd.neednet=1** kernel argument.

The following table describes how to use **ip=**, **nameserver=**, and **bond=** kernel arguments for live ISO installs.

NOTE

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

Table 1.23. Routing and bonding options for ISO

Description	Examples
To configure an IP address, either use DHCP (**ip=dhcp**) or set an individual static IP address (**ip=<host_ip>**). Then identify the DNS server IP address (**nameserver=<dns_ip>**) on each node. This example sets: • 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**	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41
Specify multiple network interfaces by specifying multiple **ip=** entries.	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
Disable DHCP on a single interface, such as when there are two or more network interfaces and only one interface is being 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
You can combine DHCP and static IP configurations on systems with multiple network interfaces.	ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none

Description	Examples
You can provide multiple DNS servers by adding a **nameserver=** entry for each server.	nameserver=1.1.1.1 nameserver=8.8.8.8
Optional: Bonding multiple network interfaces to a single interface is supported using the **bond=** option. In these two examples: • The syntax for configuring a bonded interface is: **bond=name[:network_interfaces][:options]** • *name* is the bonding device name (**bond0**), *network_interfaces* represents a comma-separated list of physical (ethernet) interfaces (**em1,em2**), 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:em1,em2: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:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none

coreos.inst boot options for ISO or PXE install

While you can pass most standard installation boot arguments to the live installer, there are several arguments that are specific to the RHCOS live installer.

- For ISO, these options can be added by interrupting the RHCOS installer.
- For PXE or iPXE, these options must be added to the **APPEND** line before starting the PXE kernel. You cannot interrupt a live PXE install.

The following table shows the RHCOS live installer boot options for ISO and PXE installs.

Table 1.24. coreos.inst boot options

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to. 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.

Argument	Description
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.
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. Once 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**.

Argument	Description
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.

coreos-installer options for ISO install

You can also install RHCOS by invoking the **coreos-installer** command directly from the command line. The kernel arguments in the previous table provide a shortcut for automatically invoking **coreos-installer** at boot time, but you can pass similar arguments directly to **coreos-installer** when running it from a shell prompt.

The following table shows the options and subcommands you can pass to the **coreos-installer** command from a shell prompt during a live install.

Table 1.25. **coreos-installer** command-line options, arguments, and subcommands

Command-line options	
Option	Description
-u, **--image-url <url>**	Specify the image URL manually.
-f, **--image-file <path>**	Specify a local image file manually.
-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.
--append-karg <arg>…	Append the default kernel argument.
--delete-karg <arg>…	Delete the default kernel argument.
-n, **--copy-network**	Copy the network configuration from the install environment.
--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.
--offline	Force offline installation.
--insecure	Skip signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Default is **x86_64**.
--preserve-on-error	Do not clear partition table on error.
-h, **--help**	Print help information.

Command-line argument

Argument	Description
<device>	The destination device.

coreos-installer embedded Ignition commands

Command	Description
$ coreos-installer iso ignition embed <options> --ignition-file <file_path> <ISO_image>	Embed an Ignition config in an ISO image.
coreos-installer iso ignition show <options> <ISO_image>	Show the embedded Ignition config from an ISO image.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.

coreos-installer ISO Ignition options

Option	Description
-f, **--force**	Overwrite an existing Ignition config.
-i, **--ignition-file <path>**	The Ignition config to be used. Default is **stdin**.
-o, **--output <path>**	Write the ISO to a new output file.
-h, **--help**	Print help information.

coreos-installer PXE Ignition commands	
Command	**Description**
Note that not all of these options are accepted by all subcommands.	
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 ignition unwrap <options> <initrd_name>	Show the wrapped Ignition config in an **initrd** image.

coreos-installer PXE Ignition options	
Option	**Description**
-i, **--ignition-file <path>**	The Ignition config to be used. Default is **stdin**.
-o, --output <path>	Write the ISO to a new output file.
-h, **--help**	Print help information.

1.3.11. Creating the cluster

To create the OpenShift Container Platform cluster, you wait for the bootstrap process to complete on the machines that you provisioned by using the Ignition config files that you generated with the installation program.

Prerequisites

- Create the required infrastructure for the cluster.
- You obtained the installation program and generated the Ignition config files for your cluster.
- You used the Ignition config files to create RHCOS machines for your cluster.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

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

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.19.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

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

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

IMPORTANT

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

1.3.12. Logging in to the cluster by using the CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

1.3.13. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.19.0
master-1  Ready     master  63m  v1.19.0
master-2  Ready     master  64m  v1.19.0
worker-0  NotReady  worker  76s  v1.19.0
worker-1  NotReady  worker  70s  v1.19.0
```

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 you approve the initial CSRs, the subsequent node client CSRs are automatically approved by the cluster **kube-controller-manager**.

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> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

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

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

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> 1
     ```

     1 **<csr_name>** is the name of a CSR from the list of current CSRs.

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

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

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

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```

NOTE

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

Additional information

- For more information on CSRs, see Certificate Signing Requests.

1.3.14. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

```
NAME                                       VERSION AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.6.0   True        False         False      3h56m
cloud-credential                           4.6.0   True        False         False      29h
cluster-autoscaler                         4.6.0   True        False         False      29h
config-operator                            4.6.0   True        False         False      6h39m
console                                    4.6.0   True        False         False      3h59m
csi-snapshot-controller                    4.6.0   True        False         False      4h12m
dns                                        4.6.0   True        False         False      4h15m
etcd                                       4.6.0   True        False         False      29h
image-registry                             4.6.0   True        False         False      3h59m
ingress                                    4.6.0   True        False         False      4h30m
insights                                   4.6.0   True        False         False      29h
kube-apiserver                             4.6.0   True        False         False      29h
kube-controller-manager                    4.6.0   True        False         False      29h
kube-scheduler                             4.6.0   True        False         False      29h
kube-storage-version-migrator              4.6.0   True        False         False      4h2m
machine-api                                4.6.0   True        False         False      29h
machine-approver                           4.6.0   True        False         False      6h34m
machine-config                             4.6.0   True        False         False      3h56m
marketplace                                4.6.0   True        False         False      4h2m
monitoring                                 4.6.0   True        False         False      6h31m
network                                    4.6.0   True        False         False      29h
node-tuning                                4.6.0   True        False         False      4h30m
openshift-apiserver                        4.6.0   True        False         False      3h56m
openshift-controller-manager               4.6.0   True        False         False      4h36m
openshift-samples                          4.6.0   True        False         False      4h30m
operator-lifecycle-manager                 4.6.0   True        False         False      29h
operator-lifecycle-manager-catalog         4.6.0   True        False         False      29h
operator-lifecycle-manager-packageserver   4.6.0   True        False         False      3h59m
service-ca                                 4.6.0   True        False         False      29h
storage                                    4.6.0   True        False         False      4h30m
```

2. Configure the Operators that are not available.

1.3.14.1. Image registry storage configuration

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

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

1.3.14.1.1. Changing the image registry's management state

To start the image registry, you must change the Image Registry Operator configuration's **managementState** from **Removed** to **Managed**.

Procedure

- Change **managementState** Image Registry Operator configuration from **Removed** to **Managed**. For example:

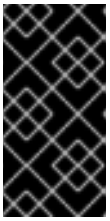
```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

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

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster that uses manually-provisioned Red Hat Enterprise Linux CoreOS (RHCOS) nodes, such as bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



IMPORTANT

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

- Must have 100Gi capacity.

Procedure

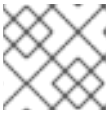
1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

NOTE

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

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

```
$ oc get pod -n openshift-image-registry
```



NOTE

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

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

Example output

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

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

4. Check the **clusteroperator** status:

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

5. 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
```

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

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

Procedure

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

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

WARNING

Configure this option for only non-production clusters.

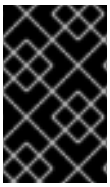
If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

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

Wait a few minutes and run the command again.

1.3.14.1.4. Configuring block registry storage

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



IMPORTANT

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

Procedure

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

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

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

3. Edit the registry configuration so that it references the correct PVC.

1.3.15. Completing installation on user-provisioned infrastructure

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

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

Procedure

1. Confirm that all the cluster components are online:

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

Example output

```
NAME                       VERSION AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication             4.6.0   True        False         False      3h56m
cloud-credential           4.6.0   True        False         False      29h
cluster-autoscaler         4.6.0   True        False         False      29h
config-operator            4.6.0   True        False         False      6h39m
console                    4.6.0   True        False         False      3h59m
csi-snapshot-controller    4.6.0   True        False         False      4h12m
```

```
dns                                        4.6.0   True        False         False      4h15m
etcd                                       4.6.0   True        False         False      29h
image-registry                             4.6.0   True        False         False      3h59m
ingress                                    4.6.0   True        False         False      4h30m
insights                                   4.6.0   True        False         False      29h
kube-apiserver                             4.6.0   True        False         False      29h
kube-controller-manager                    4.6.0   True        False         False      29h
kube-scheduler                             4.6.0   True        False         False      29h
kube-storage-version-migrator              4.6.0   True        False         False      4h2m
machine-api                                4.6.0   True        False         False      29h
machine-approver                           4.6.0   True        False         False      6h34m
machine-config                             4.6.0   True        False         False      3h56m
marketplace                                4.6.0   True        False         False      4h2m
monitoring                                 4.6.0   True        False         False      6h31m
network                                    4.6.0   True        False         False      29h
node-tuning                                4.6.0   True        False         False      4h30m
openshift-apiserver                        4.6.0   True        False         False      3h56m
openshift-controller-manager               4.6.0   True        False         False      4h36m
openshift-samples                          4.6.0   True        False         False      4h30m
operator-lifecycle-manager                 4.6.0   True        False         False      29h
operator-lifecycle-manager-catalog         4.6.0   True        False         False      29h
operator-lifecycle-manager-packageserver   4.6.0   True        False         False      3h59m
service-ca                                 4.6.0   True        False         False      29h
storage                                    4.6.0   True        False         False      4h30m
```

When all of the cluster Operators are **AVAILABLE**, you can complete the installation.

2. Monitor for cluster completion:

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

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

Example output

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

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

IMPORTANT

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

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

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

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

Example output

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

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

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

(1)

(1) Specify the pod name and namespace, as shown in the output of the previous command.

If the pod logs display, the Kubernetes API server can communicate with the cluster machines.

4. Register your cluster on the Cluster registration page.

1.3.16. Next steps

- Customize your cluster.
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.
- If necessary, you can opt out of remote health reporting .