



Installing a user-provisioned cluster on bare metal

- [Prerequisites](#)
- [Internet access for OpenShift Container Platform](#)
- [Requirements for a cluster with user-provisioned infrastructure](#)
 - [Required machines for cluster installation](#)
 - [Minimum resource requirements for cluster installation](#)
 - [Certificate signing requests management](#)
- [Requirements for baremetal clusters on vSphere](#)
 - [Networking requirements for user-provisioned infrastructure](#)
 - [User-provisioned DNS requirements](#)
 - [Load balancing requirements for user-provisioned infrastructure](#)
- [Preparing the user-provisioned infrastructure](#)
- [Validating DNS resolution for user-provisioned infrastructure](#)
- [Generating a key pair for cluster node SSH access](#)
- [Obtaining the installation program](#)
- [Installing the OpenShift CLI by downloading the binary](#)
- [Manually creating the installation configuration file](#)
 - [Sample install-config.yaml file for bare metal](#)
 - [Configuring the cluster-wide proxy during installation](#)
 - [Configuring a three-node cluster](#)
- [Creating the Kubernetes manifest and Ignition config files](#)
- [Installing RHCOS and starting the OpenShift Container Platform bootstrap process](#)
 - [Installing RHCOS by using an ISO image](#)
 - [Installing RHCOS by using PXE or iPXE booting](#)
 - [Advanced RHCOS installation configuration](#)
 - [Enabling multipathing with kernel arguments on RHCOS](#)
- [Waiting for the bootstrap process to complete](#)
- [Logging in to the cluster by using the CLI](#)
- [Approving the certificate signing requests for your machines](#)
- [Initial Operator configuration](#)
 - [Image registry removed during installation](#)
 - [Image registry storage configuration](#)
- [Completing installation on user-provisioned infrastructure](#)
- [Telemetry access for OpenShift Container Platform](#)
- [Next steps](#)

In OpenShift Container Platform 4.14, you can install a cluster on bare metal infrastructure that you provision.



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.

Prerequisites

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



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

Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.14, you require access to the internet to install your cluster.

You must have internet access to:

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



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

Additional resources

- See [Installing a user-provisioned bare metal cluster on a restricted network](#) for more information about performing a restricted network installation on bare metal infrastructure that you provision.

Requirements for a cluster with user-provisioned infrastructure

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

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

Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 1. Minimum required hosts

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



As an exception, you can run zero compute machines in a bare metal cluster that consists of three control plane machines only. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for testing, development, and production. Running one compute machine is not supported.



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

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

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

Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 2. Minimum resource requirements

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

- One CPU is equivalent to one physical core when simultaneous multithreading (SMT), or hyperthreading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = CPUs.
- OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
- As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

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

Additional resources

- [Optimizing storage](#)

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.

Additional resources

- See [Configuring a three-node cluster](#) for details about deploying three-node clusters in bare metal environments.
- See [Approving the certificate signing requests for your machines](#) for more information about approving cluster certificate signing requests after installation.

Requirements for baremetal clusters on vSphere

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

Additional resources

- See [Installing RHCOS and starting the OpenShift Container Platform bootstrap process](#) for details on setting the `disk.EnableUUID` parameter's value to `TRUE` on VMware vSphere for user-provisioned infrastructure.

Networking requirements for user-provisioned infrastructure

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require networking to be configured in `initramfs` during boot to fetch their Ignition config files.

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

It is recommended to use a DHCP server for long-term management of the cluster machines. Ensure that the DHCP server is configured to provide persistent IP addresses, DNS server information, and hostnames to the cluster machines.



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

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

Setting the cluster node hostnames through DHCP

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

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

Network connectivity requirements

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

This section provides details about the ports that are required.



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

Table 3. Ports used for all-machine to all-machine communications

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

Protocol	Port	Description
UDP	4789	VXLAN
	6081	Geneve
	9000 – 9999	Host level services, including the node exporter on ports 9100 – 9101 .
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
TCP/UDP	30000 – 32767	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 4. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	6443	Kubernetes API

Table 5. Ports used for control plane machine to control plane machine communications

Protocol	Port	Description
TCP	2379 – 2380	etcd server and peer ports

NTP configuration for user-provisioned infrastructure

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

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

Additional resources

- [Configuring chrony time service](#)

User-provisioned DNS requirements

In OpenShift Container Platform deployments, DNS name resolution is required for the following components:

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

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

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



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

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

Table 6. Required DNS records

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



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



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

Example DNS configuration for user-provisioned clusters

This section provides A and PTR record configuration samples that meet the DNS requirements for deploying OpenShift Container Platform on user-provisioned infrastructure. The samples are not meant to provide advice for choosing one DNS solution over another.

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

Example DNS A record configuration for a user-provisioned cluster

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

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.example.com.      IN A     192.168.1.5
smtp.example.com.     IN A     192.168.1.5
;
helper.example.com.   IN A     192.168.1.5
helper.ocp4.example.com. IN A     192.168.1.5
;
api.ocp4.example.com. IN A     192.168.1.5 (1)
api-int.ocp4.example.com. IN A     192.168.1.5 (2)
;
*.apps.ocp4.example.com. IN A     192.168.1.5 (3)
;
bootstrap.ocp4.example.com. IN A     192.168.1.96 (4)
;
control-plane0.ocp4.example.com. IN A     192.168.1.97 (5)
control-plane1.ocp4.example.com. IN A     192.168.1.98 (5)
control-plane2.ocp4.example.com. IN A     192.168.1.99 (5)
;
compute0.ocp4.example.com. IN A     192.168.1.11 (6)
compute1.ocp4.example.com. IN A     192.168.1.7 (6)
;
:EOF
```

- 1 Provides name resolution for the Kubernetes API. The record refers to the IP address of the API load balancer.
- 2 Provides name resolution for the Kubernetes API. The record refers to the IP address of the API load balancer and is used for internal cluster communications.
- Provides name resolution for the wildcard routes. The record refers to the IP address of the application ingress load balancer. The application ingress load balancer targets the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.

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

- 4 Provides name resolution for the bootstrap machine.
- 5 Provides name resolution for the control plane machines.
- 6 Provides name resolution for the compute machines.

Example DNS PTR record configuration for a user-provisioned cluster

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

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.

;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. (1)
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. (2)
;
96.1.168.192.in-addr.arpa. IN PTR bootstrap.ocp4.example.com. (3)
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. (4)
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com. (4)
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. (4)
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. (5)
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. (5)
;
;EOF
```

- 1 Provides reverse DNS resolution for the Kubernetes API. The PTR record refers to the record name of the API load balancer.
- 2 Provides reverse DNS resolution for the Kubernetes API. The PTR record refers to the record name of the API load balancer and is used for internal cluster communications.
- 3 Provides reverse DNS resolution for the bootstrap machine.
- 4 Provides reverse DNS resolution for the control plane machines.
- 5 Provides reverse DNS resolution for the compute machines.



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

Additional resources

- [Validating DNS resolution for user-provisioned infrastructure](#)

Load balancing requirements for user-provisioned infrastructure

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



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

The load balancing infrastructure must meet the following requirements:

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



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

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

Table 7. 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 <code>/readyz</code> 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

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

- **Application Ingress load balancer:** Provides an ingress point for application traffic flowing in from outside the cluster. A working configuration for the Ingress router is required for an OpenShift Container Platform cluster.

Configure the following conditions:

- Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.
- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.



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

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

Table 8. Application Ingress load balancer

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



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

Example load balancer configuration for user-provisioned clusters

This section provides an example API and application Ingress load balancer configuration that meets the load balancing requirements for user-provisioned clusters. The sample is an `/etc/haproxy/haproxy.cfg` configuration for an HAProxy load balancer. The example is not meant to provide advice for choosing one load balancing solution over another.

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



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

Sample API and application Ingress load balancer configuration

```
global
  log      127.0.0.1 local2
  pidfile /var/run/haproxy.pid
  maxconn 4000
  daemon
defaults
  mode          http
  log           global
  option        dontlognull
  option http-server-close
  option        redispatch
  retries       3
  timeout http-request 10s
  timeout queue   1m
  timeout connect 10s
  timeout client  1m
  timeout server  1m
  timeout http-keep-alive 10s
  timeout check   10s
  maxconn      3000
listen api-server-6443 (1)
  bind *:6443
  mode tcp
  option httpchk GET /readyz HTTP/1.0
  option log-health-checks
  balance roundrobin
  server bootstrap bootstrap.ocp4.example.com:6443 verify none check check-ssl inter 10s fall 2 rise 3
  backup (2)
    server master0 master0.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2
    rise 3
    server master1 master1.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2
    rise 3
    server master2 master2.ocp4.example.com:6443 weight 1 verify none check check-ssl inter 10s fall 2
    rise 3
listen machine-config-server-22623 (3)
  bind *:22623
  mode tcp
  server bootstrap bootstrap.ocp4.example.com:22623 check inter 1s backup (2)
  server master0 master0.ocp4.example.com:22623 check inter 1s
  server master1 master1.ocp4.example.com:22623 check inter 1s
  server master2 master2.ocp4.example.com:22623 check inter 1s
listen ingress-router-443 (4)
  bind *:443
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:443 check inter 1s
  server worker1 worker1.ocp4.example.com:443 check inter 1s
listen ingress-router-80 (5)
  bind *:80
  mode tcp
  balance source
  server worker0 worker0.ocp4.example.com:80 check inter 1s
  server worker1 worker1.ocp4.example.com:80 check inter 1s
```

- 1 Port 6443 handles the Kubernetes API traffic and points to the control plane machines.
- 2 The bootstrap entries must be in place before the OpenShift Container Platform cluster installation and they must be removed after the bootstrap process is complete.
- 3 Port 22623 handles the machine config server traffic and points to the control plane machines.

- 4 Port 443 handles the HTTPS traffic and points to the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.
- Port 80 handles the HTTP traffic and points to the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.

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



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

Preparing the user-provisioned infrastructure

Before you install OpenShift Container Platform on user-provisioned infrastructure, you must prepare the underlying infrastructure.

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

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

Prerequisites

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

Procedure

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



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

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



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

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

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

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

- Setup the required DNS infrastructure for your cluster.

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

See the *User-provisioned DNS requirements* section for more information about the OpenShift Container Platform DNS requirements.

- Validate your DNS configuration.

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

See the *Validating DNS resolution for user-provisioned infrastructure* section for detailed DNS validation steps.

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



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

Additional resources

- [Requirements for a cluster with user-provisioned infrastructure](#)
- [Installing RHCOS and starting the OpenShift Container Platform bootstrap process](#)
- [Setting the cluster node hostnames through DHCP](#)
- [Advanced RHCOS installation configuration](#)
- [Networking requirements for user-provisioned infrastructure](#)
- [User-provisioned DNS requirements](#)
- [Validating DNS resolution for user-provisioned infrastructure](#)
- [Load balancing requirements for user-provisioned infrastructure](#)

Validating DNS resolution for user-provisioned infrastructure

You can validate your DNS configuration before installing OpenShift Container Platform on user-provisioned infrastructure.



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

Prerequisites

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

Procedure

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

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

```
$ dig +noall +answer @<nameserver_ip> api.<cluster_name>.<base_domain> (1)
```

1 Replace `<nameserver_ip>` with the IP address of the nameserver, `<cluster_name>` with your cluster name, and `<base_domain>` with your base domain name.

Example output

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

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

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

Example output

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

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

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

Example output

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



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

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

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

Example output

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

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

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

Example output

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

- Use this method to perform lookups against the DNS record names for the control plane and compute nodes. Check that the results correspond to the IP addresses of each node.
- From your installation node, run reverse DNS lookups against the IP addresses of the load balancer and the cluster nodes. Validate that the record names contained in the responses correspond to the correct components.
- Perform a reverse lookup against the IP address of the API load balancer. Check that the response includes the record names for the Kubernetes API and the Kubernetes internal API:

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

Example output

5.1.168.192.in-addr.arpa. 604800	IN	PTR	api-int.ocp4.example.com. (1)
5.1.168.192.in-addr.arpa. 604800	IN	PTR	api.ocp4.example.com. (2)

1 Provides the record name for the Kubernetes internal API.

2 Provides the record name for the Kubernetes API.



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

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

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

Example output

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

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

Additional resources

- [User-provisioned DNS requirements](#)
- [Load balancing requirements for user-provisioned infrastructure](#)

Generating a key pair for cluster node SSH access

During an OpenShift Container Platform installation, you can provide an SSH public key to the installation program. The key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files and is used to authenticate SSH access to the nodes. The key is added to the `~/.ssh/authorized_keys` list for the `core` user on each node, which enables password-less authentication.

After the key is passed to the nodes, you can use the key pair to SSH in to the RHCOS nodes as the user `core`. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

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



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



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

Procedure

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

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

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



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

- View the public SSH key:

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

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

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

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



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

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

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

Example output

```
Agent pid 31874
```



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

- Add your SSH private key to the `ssh-agent`:

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

1 Specify the path and file name for your SSH private key, such as `~/.ssh/id_ed25519`

Example output

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

Next steps

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

Additional resources

- [Verifying node health](#)

Obtaining the installation program

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

Prerequisites

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

Procedure

- Access the [Infrastructure Provider](#) page on the OpenShift Cluster Manager site. If you have a Red Hat account, log in with your credentials. If you do not, create an account.
- Select your infrastructure provider.
- Navigate to the page for your installation type, download the installation program that corresponds with your host operating system and architecture, and place the file in the directory where you will store the installation configuration files.



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.

! 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.

- 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
```

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

Installing the OpenShift CLI by downloading the binary

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

! If you installed an earlier version of `oc`, you cannot use it to complete all of the commands in OpenShift Container Platform 4.14. Download and install the new version of `oc`.

Installing the OpenShift CLI on Linux

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

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the architecture from the **Product Variant** drop-down list.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.14 Linux Client** entry and save the file.
- Unpack the archive:

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

- Place the `oc` binary in a directory that is on your `PATH`.

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

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

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

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.14 Windows Client** entry and save the file.
- Unzip the archive with a ZIP program.
- 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
```

Verification

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

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

Installing the OpenShift CLI on macOS

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

Procedure

- Navigate to the [OpenShift Container Platform downloads page](#) on the Red Hat Customer Portal.
- Select the appropriate version from the **Version** drop-down list.
- Click **Download Now** next to the **OpenShift v4.14 macOS Client** entry and save the file.



For macOS arm64, choose the **OpenShift v4.14 macOS arm64 Client** entry.

- Unpack and unzip the archive.
- Move the `oc` binary to a directory on your PATH.

To check your PATH, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Manually creating the installation configuration file

Prerequisites

- You have an SSH public key on your local machine to provide to the installation program. The key will be used for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

Procedure

- Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```



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.

- Customize the sample `install-config.yaml` file template that is provided and save it in the `<installation_directory>`.



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



For some platform types, you can alternatively run `./openshift-install create install-config --dir <installation_directory>` to generate an `install-config.yaml` file. You can provide details about your cluster configuration at the prompts.

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



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

Additional resources

- [Installation configuration parameters for bare metal](#)

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: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (2)
  hyperthreading: Enabled (3)
  name: master
  replicas: 3 (5)
metadata:
  name: test (6)
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14 (7)
      hostPrefix: 23 (8)
    networkType: OVNKubernetes (9)
    serviceNetwork: (10)
    - 172.30.0.0/16
platform:
  none: {} (11)
  fips: false (12)
pullSecret: '{"auths": ...}' (13)
sshKey: 'ssh-ed25519 AAAA...' (14)
```

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

The `controlPlane` section is a single mapping, but the `compute` section is a sequence of mappings. To meet the

2 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. Only one control plane pool is used.

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

3



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



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

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

4



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

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

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

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.

7



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

8 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 ($2^{32} - 2^{23} - 2$) pod IP addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.

9 The cluster network plugin to install. The supported values are `OVNKubernetes` and `OpenShiftSDN`. The default value is `OVNKubernetes`.

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

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

11



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

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.

12



To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see [Installing the system in FIPS mode](#). When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the `x86_64`, `ppc64le`, and `s390x` architectures.

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

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

14



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.

Additional resources

- See [Load balancing requirements for user-provisioned infrastructure](#) for more information on the API and application ingress load balancing requirements.

- See [Enabling cluster capabilities](#) for more information on enabling cluster capabilities that were disabled prior to installation.
- See [Optional cluster capabilities in OpenShift Container Platform 4.14](#) for more information about the features provided by each capability.

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.



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.



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

- 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: https://<username>:<pswd>@<ip>:<port> (2)
  noProxy: example.com (3)
  additionalTrustBundle: | (4)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> (5)
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be `http`.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying.

3 Preface a domain with `.` to match subdomains only. For example, `.y.com` matches `x.y.com`, but not `y.com`. Use `*` to bypass the proxy for all destinations.

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

4 Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the `trustedCA` field of the `Proxy` object. The `additionalTrustBundle` field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

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



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



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

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

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

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



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

Configuring a three-node cluster

Optionally, you can deploy zero compute machines in a bare metal cluster that consists of three control plane machines only. This provides smaller, more resource efficient clusters for cluster administrators and developers to use for testing, development, and production.

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

Prerequisites

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

Procedure

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

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



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

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

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

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 configure the machines.

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

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

Prerequisites

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

Procedure

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

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

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



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



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

- Check that the `mastersSchedulable` parameter in the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` Kubernetes manifest file is set to `false`. This setting prevents pods from being scheduled on the control plane machines:
 - Open the `<installation_directory>/manifests/cluster-scheduler-02-config.yaml` file.
 - Locate the `mastersSchedulable` parameter and ensure that it is set to `false`.
 - Save and exit the file.
- 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.

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

```
.
  └── auth
    └── kubeadmin-password
```

```

|   └── kubeconfig
|   ├── bootstrap.ign
|   ├── master.ign
|   ├── metadata.json
|   └── worker.ign

```

Additional resources

- See [Recovering from expired control plane certificates](#) for more information about recovering kubelet certificates.

Installing RHCOS and starting the OpenShift Container Platform bootstrap process

To install OpenShift Container Platform on bare metal infrastructure that you provision, you must install Red Hat Enterprise Linux CoreOS (RHCOS) on the machines. When you install RHCOS, you must provide the Ignition config file that was generated by the OpenShift Container Platform installation program for the type of machine you are installing. If you have configured suitable networking, DNS, and load balancing infrastructure, the OpenShift Container Platform bootstrap process begins automatically after the RHCOS machines have rebooted.

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

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

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

- **Kernel arguments:** You can use kernel arguments to provide installation-specific information. For example, you can specify the locations of the RHCOS installation files that you uploaded to your HTTP server and the location of the Ignition config file for the type of node you are installing. For a PXE installation, you can use the `APPEND` parameter to pass the arguments to the kernel of the live installer. For an ISO installation, you can interrupt the live installation boot process to add the kernel arguments. In both installation cases, you can use special `coreos.inst.*` arguments to direct the live installer, as well as standard installation boot arguments for turning standard kernel services on or off.
- **Ignition configs:** OpenShift Container Platform Ignition config files (`*.ign`) are specific to the type of node you are installing. You pass the location of a bootstrap, control plane, or compute node Ignition config file during the RHCOS installation so that it takes effect on first boot. In special cases, you can create a separate, limited Ignition config to pass to the live system. That Ignition config could do a certain set of tasks, such as reporting success to a provisioning system after completing installation. This special Ignition config is consumed by the `coreos-installer` to be applied on first boot of the installed system. Do not provide the standard control plane and compute node Ignition configs to the live ISO directly.
- **coreos-installer:** You can boot the live ISO installer to a shell prompt, which allows you to prepare the permanent system in a variety of ways before first boot. In particular, you can run the `coreos-installer` command to identify various artifacts to include, work with disk partitions, and set up networking. In some cases, you can configure features on the live system and copy them to the installed system.

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.

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

Installing RHCOS by using an ISO image

You can use an ISO image to install RHCOS on the machines.

Prerequisites

- You have created the Ignition config files for your cluster.

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

Procedure

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

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

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

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



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.

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

```
$ curl -k http://<HTTP_server>/bootstrap.ign (1)
```

Example output

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

Replace `bootstrap.ign` with `master.ign` or `worker.ign` in the command to validate that the Ignition config files for the control plane and compute nodes are also available.

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

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

Example output

```
"location": "<url>/art/storage/releases/rhcos-4.14-aarch64/<release>/aarch64/rhcos-<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.14-ppc64le/<release>/ppc64le/rhcos-<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.14-s390x/<release>/s390x/rhcos-<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.14/<release>/x86_64/rhcos-<release>-live.x86_64.iso",
```



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:

`rhcoss-<version>-live.<architecture>.iso`

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



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

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

```
$ sudo coreos-installer install --ignition-url=http://<HTTP_server>/<node_type>.ign <device> --ignition-hash=sha512-<digest> (1) (2)
```

- 1 You must run the `coreos-installer` command by using `sudo`, because the `core` user does not have the required root privileges to perform the installation.

The `--ignition-hash` option is required when the Ignition config file is obtained through an HTTP URL to validate the authenticity of the Ignition config file on the cluster node. `<digest>` is the Ignition config file SHA512 digest obtained in a preceding step.



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

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

```
$ sudo coreos-installer install --ignition-url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf0116e80c59d2ea9e32ba53bc807afbc581aa059311def2c3e3b
```

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



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

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

Example command

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

- Continue to create the other machines for your cluster.



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

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

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

Installing RHCOS by using PXE or iPXE booting

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

Prerequisites

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

Procedure

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



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.

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

```
$ curl -k http://<HTTP_server>/bootstrap.ign (1)
```

Example output

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

Replace `bootstrap.ign` with `master.ign` or `worker.ign` in the command to validate that the Ignition config files for the control plane and compute nodes are also available.

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

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

Example output

```
"<url>/art/storage/releases/rhcos-4.14-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-
aarch64"
"<url>/art/storage/releases/rhcos-4.14-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.14-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.14-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-
kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.14-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.14-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.14-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.14-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.14-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.14/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.14/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.14/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```

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

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

- `kernel`: `rhcos-<version>-live-kernel-<architecture>`
- `initramfs`: `rhcos-<version>-live-initramfs.<architecture>.img`
- `rootfs`: `rhcos-<version>-live-rootfs.<architecture>.img`
- Upload the `rootfs`, `kernel`, and `initramfs` files to your HTTP server.

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

- Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
- Configure PXE or iPXE installation for the RHCOS images and begin the installation.

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

- For PXE (`x86_64`):

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
  KERNEL http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> (1)
  APPEND initrd=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img coreos.live.rootfs_1
```

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 the locations of the RHCOS files that you uploaded to your HTTP server. The `initrd` parameter value is the location of the `initramfs` file, the `coreos.live.rootfs_url` parameter value is the location of the `rootfs` file, and the `coreos.inst.ignition_url` parameter value is the location of the bootstrap Ignition config file. You can also add more kernel arguments to the `APPEND` line to configure networking or other boot options.



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

- For iPXE (x86_64 + aarch64):

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img (3)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img (3)
boot
```

Specify the locations of the RHCOS files that you uploaded to your HTTP server. The `kernel` parameter value is the location of the `kernel` file, the `initrd=main` argument is needed for booting on UEFI systems, the `coreos.live.rootfs_url` parameter value is the location of the `rootfs` file, and the `coreos.inst.ignition_url` parameter value is the location of the bootstrap Ignition config file.

- 1 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`.
- 2 Specify the location of the `initramfs` file that you uploaded to your HTTP server.



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



To network boot the CoreOS kernel on aarch64 architecture, you need to use a version of iPXE build with the `IMAGE_GZIP` option enabled. See [IMAGE_GZIP option in iPXE](#).

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

```
menuentry 'Install CoreOS' {
    linux rhcos-<version>-live-kernel-<architecture> coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-initramfs.<architecture>.img (3)
    initrd rhcos-<version>-live-initramfs.<architecture>.img (3)
}
```

Specify the locations of the RHCOS files that you uploaded to your HTTP/TFTP server. The `kernel` parameter value is the location of the `kernel` file on your TFTP server. The `coreos.live.rootfs_url` parameter value is the location of the `rootfs` file, and the `coreos.inst.ignition_url` parameter value is the location of the bootstrap Ignition config file on your HTTP Server.

- 1 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`.
- 2 Specify the location of the `initramfs` file that you uploaded to your TFTP server.

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



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

- After RHCOS installs, the system reboots. During reboot, the system applies the Ignition config file that you specified.
- Check the console output to verify that Ignition ran.

Example command

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

- Continue to create the machines for your cluster.



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

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



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

Advanced 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
- Customizing a live ISO or PXE boot image

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

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

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

```
$ sudo coreos-installer install --copy-network \
--ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```



The `--copy-network` option only copies networking configuration found under `/etc/NetworkManager/system-connections`. In particular, it does not copy the system hostname.

- Reboot into the installed system.

Additional resources

- See [Getting started with nmcli](#) and [Getting started with nmtui](#) in the RHEL 8 documentation for more information about the `nmcli` and `nmtui` tools.

Disk partitioning

The disk partitions are created on OpenShift Container Platform cluster nodes during the Red Hat Enterprise Linux CoreOS (RHCOS) installation. Each RHCOS node of a particular architecture uses the same partition layout, unless the default partitioning configuration is overridden. During the RHCOS installation, the size of the root file system is increased to use the remaining available space on the target device.

There are two cases where you might want to override the default partitioning when installing RHCOS on an OpenShift Container Platform cluster node:

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



For disk sizes larger than 100GB, and especially disk sizes larger than 1TB, create a separate `/var` partition. See "Creating a separate `/var` partition" and this [Red Hat Knowledgebase article](#) for more information.



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

- **Retaining 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.

A

The use of custom partitions could result in those partitions not being monitored by OpenShift Container Platform or alerted on. If you are overriding the default partitioning, see [Understanding OpenShift File System Monitoring \(eviction conditions\)](#) for more information about how OpenShift Container Platform monitors your host file systems.

Creating a separate `/var` partition

In general, you should use the default disk partitioning that is created during the RHCOS installation. However, there are cases where you might want to create a separate partition for a directory that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the `/var` directory 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.



For disk sizes larger than 100GB, and especially larger than 1TB, create a separate `/var` partition.

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.

The use of a separate partition for the `/var` directory or a subdirectory of `/var` also prevents data growth in the partitioned directory from filling up the root file system.

The following procedure sets up a separate `/var` partition by adding a machine config manifest that is wrapped into the Ignition config file for a node type during the preparation phase of an installation.

Procedure

- On your installation host, change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

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

- Create a Butane config that configures the additional partition. For example, name the file `$HOME/clusterconfig/98-var-partition.bu`, change the disk device name to the name of the storage device on the `worker` systems, and set the storage size as appropriate. This example places the `/var` directory on a separate partition:

```
variant: openshift
version: 4.14.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
    - device: /dev/disk/by-id/<device_name> (1)
      partitions:
        - label: var
          start_mib: <partition_start_offset> (2)
          size_mib: <partition_size> (3)
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] (4)
      with_mount_unit: true
```

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

When adding a data partition to the boot disk, a minimum offset 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 offset value is specified, or if

- 2 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.

- 3 The size of the data partition in mebibytes.

- 4 The `prjquota` mount option must be enabled for filesystems used for container storage.



When creating a separate `/var` partition, you cannot use different instance types for compute nodes, if the different instance types do not have the same device name.

- Create a manifest from the Butane config and save it to the `clusterconfig/openshift` directory. For example, run the following command:

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

- Create the Ignition config files:

```
$ openshift-install create ignition-configs --dir <installation_directory> (1)
```

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

Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory:

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

The files in the `<installation_directory>/manifest` and `<installation_directory>/openshift` directories are wrapped into the Ignition config files, including the file that contains the `98-var-partition` custom `MachineConfig` object.

Next steps

- You can apply the custom disk partitioning by referencing the Ignition config files during the RHCOS installations.

Retaining existing partitions

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

Saved partitions might be data partitions from an existing OpenShift Container Platform system. You can identify the disk partitions you want to keep either by partition label or by number.



If you save existing partitions, and those partitions do not leave enough space for RHCOS, the installation will fail without damaging the saved partitions.

Retaining existing partitions during 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/disk/by-id/scsi-<serial_number>
```

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/disk/by-id/scsi-<serial_number>
```

This example preserves partitions 5 and higher:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
--save-partindex 5- /dev/disk/by-id/scsi-<serial_number>
```

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

Retaining existing partitions during 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
```

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.



It is not recommended to modify these Ignition config files directly. You can update the manifest files that are wrapped into the Ignition config files, as outlined in examples in the preceding sections.

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 can be created by using the `coreos-installer customize` subcommand and its various options. With this method, the Ignition config passes to the live install medium, runs immediately upon booting, and performs setup tasks before or after the RHCOS system installs to disk. This method should only be used for performing tasks that must be done 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.

Default console configuration

Red Hat Enterprise Linux CoreOS (RHCOS) nodes installed from an OpenShift Container Platform 4.14 boot image use a default console that is meant to accommodate most virtualized and bare metal setups. Different cloud and virtualization platforms may use different default settings depending on the chosen architecture. Bare metal installations use the kernel default settings which typically means the graphical console is the primary console and the serial console is disabled.

The default consoles may not match your specific hardware configuration or you might have specific needs that require you to adjust the default console. For example:

- You want to access the emergency shell on the console for debugging purposes.
- Your cloud platform does not provide interactive access to the graphical console, but provides a serial console.
- You want to enable multiple consoles.

Console configuration is inherited from the boot image. This means that new nodes in existing clusters are unaffected by changes to the default console.

You can configure the console for bare metal installations in the following ways:

- Using `coreos-installer` manually on the command line.
- Using the `coreos-installer iso customize` or `coreos-installer pxe customize` subcommands with the `--dest-console` option to create a custom image that automates the process.



For advanced customization, perform console configuration using the `coreos-installer iso` or `coreos-installer pxe` subcommands, and not kernel arguments.

Enabling the serial console for PXE and ISO installations

By default, the Red Hat Enterprise Linux CoreOS (RHCOS) serial console is disabled and all output is written to the graphical console. You can enable the serial console for an ISO installation and reconfigure the bootloader so that output is sent to both the serial console and the graphical console.

Procedure

- Boot the ISO installer.
- Run the `coreos-installer` command to install the system, adding the `--console` option once to specify the graphical console, and a second time to specify the serial console:

```
$ coreos-installer install \
--console=tty0 \ (1)
--console=ttyS0,<options> \ (2)
--ignition-url=http://host/worker.ign /dev/disk/by-id/scsi-<serial_number>
```

- 1 The desired secondary console. In this case, the graphical console. Omitting this option will disable the graphical console.

The desired primary console. In this case the serial console. The `options` field defines the baud rate and other settings.

- 2 A common value for this field is `11520n8`. If no options are provided, the default kernel value of `9600n8` is used. For more information on the format of this option, see [Linux kernel serial console](#) documentation.

- Reboot into the installed system.



A similar outcome can be obtained by using the `coreos-installer install --append-karg` option, and specifying the console with `console=`. However, this will only set the console for the kernel and not the bootloader.

To configure a PXE installation, make sure the `coreos.inst.install_dev` kernel command line option is omitted, and use the shell prompt to run `coreos-installer` manually using the above ISO installation procedure.

Customizing a live RHCOS ISO or PXE install

You can use the live ISO image or PXE environment to install RHCOS by injecting an Ignition config file directly into the image. This creates a customized image that you can use to provision your system.

For an ISO image, the mechanism to do this is the `coreos-installer iso customize` subcommand, which modifies the `.iso` file with your configuration. Similarly, the mechanism for a PXE environment is the `coreos-installer pxe customize` subcommand, which creates a new `initramfs` file that includes your customizations.

The `customize` subcommand is a general purpose tool that can embed other types of customizations as well. The following tasks are examples of some of the more common customizations:

- Inject custom CA certificates for when corporate security policy requires their use.
- Configure network settings without the need for kernel arguments.
- Embed arbitrary preinstall and post-install scripts or binaries.

Customizing a live RHCOS ISO image

You can customize a live RHCOS ISO image directly with the `coreos-installer iso customize` subcommand. When you boot the ISO image, the customizations are applied automatically.

You can use this feature to configure the ISO image to automatically install RHCOS.

Procedure

- Download the `coreos-installer` binary from the [coreos-installer image mirror](#) page.
- Retrieve the RHCOS ISO image from the [RHCOS image mirror](#) page and the Ignition config file, and then run the following command to inject the Ignition config directly into the ISO image:

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
--dest-ignition bootstrap.ign \ (1)
--dest-device /dev/disk/by-id/scsi-<serial_number> (2)
```

- 1 The Ignition config file that is generated from the `openshift-installer` installation program.

When you specify this option, the ISO image automatically runs an installation. Otherwise, the image remains

- 2 configured for installation, but does not install automatically unless you specify the `coreos.inst.install_dev` kernel argument.

- Optional: To remove the ISO image customizations and return the image to its pristine state, run:

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

You can now re-customize the live ISO image or use it in its pristine state.

Applying your customizations affects every subsequent boot of RHCOS.

Modifying a live install ISO image to enable the serial console

On clusters installed with OpenShift Container Platform 4.12 and above, the serial console is disabled by default and all output is written to the graphical console. You can enable the serial console with the following procedure.

Procedure

- Download the `coreos-installer` binary from the [coreos-installer image mirror](#) page.
- Retrieve the RHCOS ISO image from the [RHCOS image mirror](#) page and run the following command to customize the ISO image to enable the serial console to receive output:

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
--dest-ignition <path> \ (1)
--dest-console tty0 \ (2)
--dest-console ttys0,<options> \ (3)
--dest-device /dev/disk/by-id/scsi-<serial_number> \ (4)
```

1 The location of the Ignition config to install.

2 The desired secondary console. In this case, the graphical console. Omitting this option will disable the graphical console.

3 The desired primary console. In this case, the serial console. The `options` field defines the baud rate and other settings. A common value for this field is `115200n8`. If no options are provided, the default kernel value of `9600n8` is used. For more information on the format of this option, see the [Linux kernel serial console](#) documentation.

4 The specified disk to install to. If you omit this option, the ISO image automatically runs the installation program which will fail unless you also specify the `coreos.inst.install_dev` kernel argument.



The `--dest-console` option affects the installed system and not the live ISO system. To modify the console for a live ISO system, use the `--live-karg-append` option and specify the console with `console=`.

Your customizations are applied and affect every subsequent boot of the ISO image.

- Optional: To remove the ISO image customizations and return the image to its original state, run the following command:

```
$ coreos-installer iso reset rhcos-<version>-live.x86_64.iso
```

You can now recustomize the live ISO image or use it in its original state.

Modifying a live install ISO image to use a custom certificate authority

You can provide certificate authority (CA) certificates to Ignition with the `--ignition-ca` flag of the `customize` subcommand. You can use the CA certificates during both the installation boot and when provisioning the installed system.



Custom CA certificates affect how Ignition fetches remote resources but they do not affect the certificates installed onto the system.

Procedure

- Download the `coreos-installer` binary from the [coreos-installer image mirror](#) page.
- Retrieve the RHCOS ISO image from the [RHCOS image mirror](#) page and run the following command to customize the ISO image for use with a custom CA:

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso --ignition-ca cert.pem
```



The `coreos.inst.ignition_url` kernel parameter does not work with the `--ignition-ca` flag. You must use the `--dest-ignition` flag to create a customized image for each cluster.

Applying your custom CA certificate affects every subsequent boot of RHCOS.

Modifying a live install ISO image with customized network settings

You can embed a NetworkManager keyfile into the live ISO image and pass it through to the installed system with the `--network-keyfile` flag of the `customize` subcommand.



When creating a connection profile, you must use a `.nmconnection` filename extension in the filename of the connection profile. If you do not use a `.nmconnection` filename extension, the cluster will apply the connection profile to the live environment, but it will not apply the configuration when the cluster first boots up the nodes, resulting in a setup that does not work.

Procedure

- Download the `coreos-installer` binary from the [coreos-installer image mirror](#) page.
- Create a connection profile for a bonded interface. For example, create the `bond0.nmconnection` file in your local directory with the following content:

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

- Create a connection profile for a secondary interface to add to the bond. For example, create the `bond0-proxy-em1.nmconnection` file in your local directory with the following content:

```
[connection]
id=em1
type=ethernet
interface-name=em1
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

- Create a connection profile for a secondary interface to add to the bond. For example, create the `bond0-proxy-em2.nmconnection` file in your local directory with the following content:

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

- Retrieve the RHCOS ISO image from the [RHCOS image mirror](#) page and run the following command to customize the ISO image with your configured networking:

```
$ coreos-installer iso customize rhcos-<version>-live.x86_64.iso \
--network-keyfile bond0.nmconnection \
--network-keyfile bond0-proxy-em1.nmconnection \
--network-keyfile bond0-proxy-em2.nmconnection
```

Network settings are applied to the live system and are carried over to the destination system.

Customizing a live RHCOS PXE environment

You can customize a live RHCOS PXE environment directly with the `coreos-installer pxe customize` subcommand. When you boot the PXE environment, the customizations are applied automatically.

You can use this feature to configure the PXE environment to automatically install RHCOS.

Procedure

- Download the `coreos-installer` binary from the [coreos-installer image mirror](#) page.
- Retrieve the RHCOS kernel, `initramfs` and `rootfs` files from the [RHCOS image mirror](#) page and the Ignition config file, and then run the following command to create a new `initramfs` file that contains the customizations from your Ignition config:

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
--dest-ignition bootstrap.ign \ (1)
--dest-device /dev/disk/by-id/scsi-<serial_number> \ (2)
-o rhcos-<version>-custom-initramfs.x86_64.img (3)
```

1 The Ignition config file that is generated from `openshift-installer`.

When you specify this option, the PXE environment automatically runs an install. Otherwise, the image remains 2 configured for installing, but does not do so automatically unless you specify the `coreos.inst.install_dev` kernel argument.

3 Use the customized `initramfs` file in your PXE configuration. Add the `ignition.firstboot` and `ignition.platform.id=metal` kernel arguments if they are not already present.

Applying your customizations affects every subsequent boot of RHCOS.

Modifying a live install PXE environment to enable the serial console

On clusters installed with OpenShift Container Platform 4.12 and above, the serial console is disabled by default and all output is written to the graphical console. You can enable the serial console with the following procedure.

Procedure

- Download the `coreos-installer` binary from the [coreos-installer image mirror](#) page.
- Retrieve the RHCOS kernel, `initramfs` and `rootfs` files from the [RHCOS image mirror](#) page and the Ignition config file, and then run the following command to create a new customized `initramfs` file that enables the serial console to receive output:

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
--dest-ignition <path> \ (1)
--dest-console tty0 \ (2)
--dest-console ttyS0,<options> \ (3)
--dest-device /dev/disk/by-id/scsi-<serial_number> \ (4)
-o rhcos-<version>-custom-initramfs.x86_64.img \ (5)
```

1 The location of the Ignition config to install.

2 The desired secondary console. In this case, the graphical console. Omitting this option will disable the graphical console.

3 The desired primary console. In this case, the serial console. The options field defines the baud rate and other settings. A common value for this field is 115200n8 . If no options are provided, the default kernel value of 9600n8 is used. For more information on the format of this option, see the [Linux kernel serial console documentation](#).

4 The specified disk to install to. If you omit this option, the PXE environment automatically runs the installer which will fail unless you also specify the coreos.inst.install_dev kernel argument.

5 Use the customized initramfs file in your PXE configuration. Add the ignition.firstboot and ignition.platform.id=metal kernel arguments if they are not already present.

Your customizations are applied and affect every subsequent boot of the PXE environment.

Modifying a live install PXE environment to use a custom certificate authority

You can provide certificate authority (CA) certificates to Ignition with the --ignition-ca flag of the customize subcommand.

You can use the CA certificates during both the installation boot and when provisioning the installed system.



Custom CA certificates affect how Ignition fetches remote resources but they do not affect the certificates installed onto the system.

Procedure

- Download the coreos-installer binary from the [coreos-installer image mirror](#) page.
- Retrieve the RHCOS kernel, initramfs and rootfs files from the [RHCOS image mirror](#) page and run the following command to create a new customized initramfs file for use with a custom CA:

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
--ignition-ca cert.pem \
-o rhcos-<version>-custom-initramfs.x86_64.img
```

- Use the customized initramfs file in your PXE configuration. Add the ignition.firstboot and ignition.platform.id=metal kernel arguments if they are not already present.



The coreos.inst.ignition_url kernel parameter does not work with the --ignition-ca flag. You must use the --dest-ignition flag to create a customized image for each cluster.

Applying your custom CA certificate affects every subsequent boot of RHCOS.

Modifying a live install PXE environment with customized network settings

You can embed a NetworkManager keyfile into the live PXE environment and pass it through to the installed system with the --network-keyfile flag of the customize subcommand.



When creating a connection profile, you must use a .nmconnection filename extension in the filename of the connection profile. If you do not use a .nmconnection filename extension, the cluster will apply the connection profile to the live environment, but it will not apply the configuration when the cluster first boots up the nodes, resulting in a setup that does not work.

Procedure

- Download the coreos-installer binary from the [coreos-installer image mirror](#) page.
- Create a connection profile for a bonded interface. For example, create the bond0.nmconnection file in your local directory with the following content:

```
[connection]
id=bond0
type=bond
interface-name=bond0
multi-connect=1
permissions=

[ethernet]
mac-address-blacklist=

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto

[proxy]
```

- Create a connection profile for a secondary interface to add to the bond. For example, create the `bond0-proxy-em1.nmconnection` file in your local directory with the following content:

```
[connection]
id=em1
type=ethernet
interface-name=em1
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

- Create a connection profile for a secondary interface to add to the bond. For example, create the `bond0-proxy-em2.nmconnection` file in your local directory with the following content:

```
[connection]
id=em2
type=ethernet
interface-name=em2
master=bond0
multi-connect=1
permissions=
slave-type=bond

[ethernet]
mac-address-blacklist=
```

- Retrieve the RHCOS kernel, initramfs and rootfs files from the [RHCOS image mirror](#) page and run the following command to create a new customized initramfs file that contains your configured networking:

```
$ coreos-installer pxe customize rhcos-<version>-live-initramfs.x86_64.img \
--network-keyfile bond0.nmconnection \
--network-keyfile bond0-proxy-em1.nmconnection \
--network-keyfile bond0-proxy-em2.nmconnection \
-o rhcos-<version>-custom-initramfs.x86_64.img
```

- Use the customized initramfs file in your PXE configuration. Add the `ignition.firstboot` and `ignition.platform.id=metal` kernel arguments if they are not already present.

Network settings are applied to the live system and are carried over to the destination system.

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.

Networking and bonding options for ISO installations

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



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

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



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

The networking options are passed to the `dracut` tool during system boot. For more information about the networking options supported by `dracut`, see the [dracut cmdline manual page](#).

The following examples are the networking options for ISO installation.

Configuring DHCP or static IP addresses

To configure an IP address, either use DHCP (`ip=dhcp`) or set an individual static IP address (`ip=<host_ip>`). If setting a static IP, you must then identify the DNS server IP address (`nameserver=<dns_ip>`) on each node. The following 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`
- The auto-configuration value to `none`. No auto-configuration is required when IP networking is configured statically.

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



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

Configuring an IP address without a static hostname

You can configure an IP address without assigning a static hostname. If a static hostname is not set by the user, it will be picked up and automatically set by a reverse DNS lookup. To configure an IP address without a static hostname refer to the following example:

- The node's IP address to `10.10.10.2`
- The gateway address to `10.10.10.254`
- The netmask to `255.255.255.0`

- The DNS server address to 4.4.4.41
- The auto-configuration value to none. No auto-configuration is required when IP networking is configured statically.

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

Specifying multiple network interfaces

You can specify multiple network interfaces by setting 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
```

Configuring default gateway and route

Optional: You can configure routes to additional networks by setting an rd.route= value.



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

- Run the following command to configure the default gateway:

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

- Enter the following command to configure the route for the additional network:

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

Disabling DHCP on a single interface

You can disable DHCP on a single interface, such as when there are two or more network interfaces and only one interface is being used. In the example, the enp1s0 interface has a static networking configuration and DHCP is disabled for enp2s0, which is not used:

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

Combining DHCP and static IP configurations

You can combine DHCP and static IP configurations on systems with multiple network interfaces, for example:

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

Configuring VLANs on individual interfaces

Optional: You can configure VLANs on individual interfaces by using the vlan= parameter.

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

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

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

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

Providing multiple DNS servers

You can provide multiple DNS servers by adding a nameserver= entry for each server, for example:

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

Bonding multiple network interfaces to a single interface

Optional: You can bond multiple network interfaces to a single interface by using the `bond=` option. Refer to the following 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
```

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



Support for Day 1 operations associated with enabling NIC partitioning for SR-IOV devices is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

Optional: You can bond multiple SR-IOV network interfaces to a dual port NIC interface by using the `bond=` option.

On each node, you must perform the following tasks:

- Create the SR-IOV virtual functions (VFs) following the guidance in [Managing SR-IOV devices](#). Follow the procedure in the "Attaching SR-IOV networking devices to virtual machines" section.
- Create the bond, attach the desired VFs to the bond and set the bond link state up following the guidance in [Configuring network bonding](#). Follow any of the described procedures to create the bond.

The following examples illustrate the syntax you must use:

- The syntax for configuring a bonded interface is `bond=<name>[:<network_interfaces>][:options]`.
- `<name>` is the bonding device name (`bond0`), `<network_interfaces>` represents the virtual functions (VFs) by their known name in the kernel and shown in the output of the `ip link` command (`eno1f0, eno2f0`), and `options` is a comma-separated list of bonding options. Enter `modinfo bonding` to see available options.
- When you create a bonded interface using `bond=`, you must specify how the IP address is assigned and other information for the bonded interface.
 - To configure the bonded interface to use DHCP, set the bond's IP address to `dhcp`. For example:

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

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

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

Using network teaming

Optional: You can use a network teaming as an alternative to bonding by using the `team=` parameter:

- The syntax for configuring a team interface is: `team=name[:network_interfaces]`

`name` is the team device name (`team0`) and `network_interfaces` represents a comma-separated list of physical (ethernet) interfaces (`em1`, `em2`).



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

Use the following example to configure a network team:

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

coreos-installer options for ISO and PXE installations

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

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

Table 9. `coreos-installer` subcommands, command-line options, and arguments

coreos-installer install subcommand

Subcommand	Description
<code>\$ coreos-installer install <options> <device></code>	Embed an Ignition config in an ISO image.

coreos-installer install subcommand options

Option	Description
<code>-u, --image-url <url></code>	Specify the image URL manually.
<code>-f, --image-file <path></code>	Specify a local image file manually. Used for debugging.
<code>-i, --ignition-file <path></code>	Embed an Ignition config from a file.
<code>-I, --ignition-url <URL></code>	Embed an Ignition config from a URL.
<code>--ignition-hash <digest></code>	Digest <code>type-value</code> of the Ignition config.
<code>-p, --platform <name></code>	Override the Ignition platform ID for the installed system.
<code>--console <spec></code>	Set the kernel and bootloader console for the installed system. For more information about the format of <code><spec></code> , see the Linux kernel serial console documentation.

--append-karg <arg>...	Append a default kernel argument to the installed system.
--delete-karg <arg>...	Delete a default kernel argument from the installed system.
-n, --copy-network	Copy the network configuration from the install environment.
	<p style="text-align: center;">!</p> <p>The <code>--copy-network</code> option only copies networking configuration found under <code>/etc/NetworkManager/system-connections</code>. In particular, it does not copy the system hostname.</p>
--network-dir <path>	For use with <code>-n</code> . Default is <code>/etc/NetworkManager/system-connections/</code> .
--save-partlabel <lx>..	Save partitions with this label glob.
--save-partindex <id>..	Save partitions with this number or range.
--insecure	Skip RHCOS image signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Valid values are <code>x86_64</code> and <code>aarch64</code> .
--preserve-on-error	Do not clear partition table on error.
-h, --help	Print help information.
coreos-installer install subcommand argument	
<i>Argument</i>	<i>Description</i>
<device>	The destination device.
coreos-installer ISO subcommands	
<i>Subcommand</i>	<i>Description</i>
\$ coreos-installer iso customize <options> <ISO_image>	Customize a RHCOS live ISO image.
coreos-installer iso reset <options> <ISO_image>	Restore a RHCOS live ISO image to default settings.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.
coreos-installer ISO customize subcommand options	
<i>Option</i>	<i>Description</i>
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and bootloader console for the destination system.
--dest-device <path>	Install and overwrite the specified destination device.
--dest-karg-append <arg>	Add a kernel argument to each boot of the destination system.
--dest-karg-delete <arg>	Delete a kernel argument from each boot of the destination system.

--network-keyfile <path>	Configure networking by using the specified NetworkManager keyfile for live and destination systems.
--ignition-ca <path>	Specify an additional TLS certificate authority to be trusted by Ignition.
--pre-install <path>	Run the specified script before installation.
--post-install <path>	Run the specified script after installation.
--installer-config <path>	Apply the specified installer configuration file.
--live-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
--live-karg-append <arg>	Add a kernel argument to each boot of the live environment.
--live-karg-delete <arg>	Delete a kernel argument from each boot of the live environment.
--live-karg-replace <k=o=n>	Replace a kernel argument in each boot of the live environment, in the form key=old=new .
-f, --force	Overwrite an existing Ignition config.
-o, --output <path>	Write the ISO to a new output file.
-h, --help	Print help information.

coreos-installer PXE subcommands

<i>Subcommand</i>	<i>Description</i>
Note that not all of these options are accepted by all subcommands.	
coreos-installer pxe customize <options> <path>	Customize a RHCOS live PXE boot config.
coreos-installer pxe ignition wrap <options>	Wrap an Ignition config in an image.
coreos-installer pxe ignition unwrap <options> <image_name>	Show the wrapped Ignition config in an image.

coreos-installer PXE customize subcommand options

<i>Option</i>	<i>Description</i>
Note that not all of these options are accepted by all subcommands.	
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and bootloader console for the destination system.
--dest-device <path>	Install and overwrite the specified destination device.
--network-keyfile <path>	Configure networking by using the specified NetworkManager keyfile for live and destination systems.
--ignition-ca <path>	Specify an additional TLS certificate authority to be trusted by Ignition.
--pre-install <path>	Run the specified script before installation.

<code>post-install <path></code>	Run the specified script after installation.
<code>--installer-config <path></code>	Apply the specified installer configuration file.
<code>--live-ignition <path></code>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
<code>-o, --output <path></code>	Write the initramfs to a new output file.
	 This option is required for PXE environments.
<code>-h, --help</code>	Print help information.

coreos.inst boot options for ISO or PXE installations

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

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

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

Table 10. `coreos.inst` boot options

Argument	Description
<code>coreos.inst.install_dev</code>	Required. The block device on the system to install to. It is recommended to use the full path, such as <code>/dev/sda</code> , although <code>sda</code> is allowed.
<code>coreos.inst.ignition_url</code>	Optional: The URL of the Ignition config to embed into the installed system. If no URL is specified, no Ignition config is embedded. Only HTTP and HTTPS protocols are supported.
<code>coreos.inst.save_partlabel</code>	Optional: Comma-separated labels of partitions to preserve during the install. Glob-style wildcards are permitted. The specified partitions do not need to exist.
<code>coreos.inst.save_partindex</code>	Optional: Comma-separated indexes of partitions to preserve during the install. Ranges <code>m-n</code> are permitted, and either <code>m</code> or <code>n</code> can be omitted. The specified partitions do not need to exist.
<code>coreos.inst.insecure</code>	Optional: Permits the OS image that is specified by <code>coreos.inst.image_url</code> to be unsigned.
<code>coreos.inst.image_url</code>	Optional: Download and install the specified RHCOS image. <ul style="list-style-type: none"> 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 <code>coreos.inst.image_url</code>, you must also use <code>coreos.inst.insecure</code>. This is because the bare-metal media are not GPG-signed for OpenShift Container Platform. Only HTTP and HTTPS protocols are supported.

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

Enabling multipathing with kernel arguments on RHCOS

RHCOS supports multipathing on the primary disk, allowing stronger resilience to hardware failure to achieve higher host availability.

You can enable multipathing at installation time for nodes that were provisioned in OpenShift Container Platform 4.8 or later. While postinstallation support is available by activating multipathing via the machine config, enabling multipathing during installation is recommended.

In setups where any I/O to non-optimized paths results in I/O system errors, you must enable multipathing at installation time.



On IBM Z® and IBM® LinuxONE, you can enable multipathing only if you configured your cluster for it during installation. For more information, see "Installing RHCOS and starting the OpenShift Container Platform bootstrap process" in *Installing a cluster with z/VM on IBM Z® and IBM® LinuxONE*.

The following procedure enables multipath at installation time and appends kernel arguments to the `coreos-installer install` command so that the installed system itself will use multipath beginning from the first boot.



OpenShift Container Platform does not support enabling multipathing as a day-2 activity on nodes that have been upgraded from 4.6 or earlier.

Procedure

- To enable multipath and start the `multipathd` daemon, run the following command:

```
$ mpathconf --enable && systemctl start multipathd.service
```

- Optional: If booting the PXE or ISO, you can instead enable multipath by adding `rd.multipath=default` from the kernel command line.
- Append the kernel arguments by invoking the `coreos-installer` program:
 - If there is only one multipath device connected to the machine, it should be available at path `/dev/mapper/mpatha`. For example:

```
$ coreos-installer install /dev/mapper/mpatha \ (1)
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1 Indicates the path of the single multipathed device.

- If there are multiple multipath devices connected to the machine, or to be more explicit, instead of using `/dev/mapper/mpatha`, it is recommended to use the World Wide Name (WWN) symlink available in `/dev/disk/by-id`. For example:

```
$ coreos-installer install /dev/disk/by-id/wwn-<wwn_ID> \ (1)
--append-karg rd.multipath=default \
--append-karg root=/dev/disk/by-label/dm-mpath-root \
--append-karg rw
```

1 Indicates the WWN ID of the target multipathed device. For example, `0xx194e957fcedb4841`.

This symlink can also be used as the `coreos.inst.install_dev` kernel argument when using special `coreos.inst.*` arguments to direct the live installer. For more information, see "Installing RHCOS and starting the OpenShift Container Platform bootstrap process".

- Check that the kernel arguments worked by going to one of the worker nodes and listing the kernel command line arguments (in `/proc/cmdline` on the host):

```
$ oc debug node/ip-10-0-141-105.ec2.internal
```

Example output

```
Starting pod/ip-10-0-141-105ec2internal-debug ...
To use host binaries, run `chroot /host`

sh-4.2# cat /host/proc/cmdline
...
rd.multipath=default root=/dev/disk/by-label/dm-mpath-root
...
sh-4.2# exit
```

You should see the added kernel arguments.

Additional resources

- See [Installing RHCOS and starting the OpenShift Container Platform bootstrap process](#) for more information on using special `coreos.inst.*` arguments to direct the live installer.

Waiting for the bootstrap process to complete

The OpenShift Container Platform bootstrap process begins after the cluster nodes first boot into the persistent RHCOS environment that has been installed to disk. The configuration information provided through the Ignition config files is used to initialize the bootstrap process and install OpenShift Container Platform on the machines. You must wait for the bootstrap process to complete.

Prerequisites

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

Procedure

- 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 v1.27.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

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

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



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

Additional resources

- See [Monitoring installation progress](#) for more information about monitoring the installation logs and retrieving diagnostic data if installation issues arise.

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

- 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.

- Verify you can run `oc` commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

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

- Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
master-0	Ready	master	63m	v1.27.3
master-1	Ready	master	63m	v1.27.3
master-2	Ready	master	64m	v1.27.3

The output lists all of the machines that you created.



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

- 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.

- 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:



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



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
```



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

- 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
CSR-bfd72	5m26s	system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending		
CSR-c571v	5m26s	system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending		
...		

- 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
```

- 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	v1.27.3
master-1	Ready	master	73m	v1.27.3
master-2	Ready	master	74m	v1.27.3
worker-0	Ready	worker	11m	v1.27.3
worker-1	Ready	worker	11m	v1.27.3



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](#).

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

- Watch the cluster components come online:

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

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.14.0	True	False	False	19m
baremetal	4.14.0	True	False	False	37m
cloud-credential	4.14.0	True	False	False	40m
cluster-autoscaler	4.14.0	True	False	False	37m
config-operator	4.14.0	True	False	False	38m
console	4.14.0	True	False	False	26m
csi-snapshot-controller	4.14.0	True	False	False	37m
dns	4.14.0	True	False	False	37m
etcd	4.14.0	True	False	False	36m
image-registry	4.14.0	True	False	False	31m
ingress	4.14.0	True	False	False	30m
insights	4.14.0	True	False	False	31m
kube-apiserver	4.14.0	True	False	False	26m
kube-controller-manager	4.14.0	True	False	False	36m
kube-scheduler	4.14.0	True	False	False	36m
kube-storage-version-migrator	4.14.0	True	False	False	37m
machine-api	4.14.0	True	False	False	29m
machine approver	4.14.0	True	False	False	37m
machine-config	4.14.0	True	False	False	36m
marketplace	4.14.0	True	False	False	37m
monitoring	4.14.0	True	False	False	29m
network	4.14.0	True	False	False	38m
node-tuning	4.14.0	True	False	False	37m
openshift-apiserver	4.14.0	True	False	False	32m
openshift-controller-manager	4.14.0	True	False	False	30m
openshift-samples	4.14.0	True	False	False	32m
operator-lifecycle-manager	4.14.0	True	False	False	37m
operator-lifecycle-manager-catalog	4.14.0	True	False	False	37m
operator-lifecycle-manager-packageserver	4.14.0	True	False	False	32m
service-ca	4.14.0	True	False	False	38m
storage	4.14.0	True	False	False	37m

- Configure the Operators that are not available.

Additional resources

- See [Gathering logs from a failed installation](#) for details about gathering data in the event of a failed OpenShift Container Platform installation.
- See [Troubleshooting Operator issues](#) for steps to check Operator pod health across the cluster and gather Operator logs for diagnosis.

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`.

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.

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

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



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

- Must have 100Gi capacity.

Procedure

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



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

- Verify that you do not have a registry pod:

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

Example output

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



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

- 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.

- Check the `clusteroperator` status:

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

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE	MESSAGE
image-registry	4.14	True	False	False	6h50m	

- 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
```

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":{}}}}'
```



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.

Configuring block registry storage for bare metal

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



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

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

Procedure

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

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

- 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.

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

1 A unique name that represents the `PersistentVolumeClaim` object.

2 The namespace for the `PersistentVolumeClaim` object, which is `openshift-image-registry`.

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

4 The size of the persistent volume claim.

- Enter the following command to create the `PersistentVolumeClaim` object from the file:

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

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

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

Example output

```
storage:
pvc:
claim: (1)
```

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

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

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

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

Example output

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.14.0	True	False	False	19m
baremetal	4.14.0	True	False	False	37m
cloud-credential	4.14.0	True	False	False	40m
cluster-autoscaler	4.14.0	True	False	False	37m
config-operator	4.14.0	True	False	False	38m
console	4.14.0	True	False	False	26m
csi-snapshot-controller	4.14.0	True	False	False	37m
dns	4.14.0	True	False	False	37m
etcd	4.14.0	True	False	False	36m
image-registry	4.14.0	True	False	False	31m
ingress	4.14.0	True	False	False	30m
insights	4.14.0	True	False	False	31m
kube-apiserver	4.14.0	True	False	False	26m
kube-controller-manager	4.14.0	True	False	False	36m
kube-scheduler	4.14.0	True	False	False	36m
kube-storage-version-migrator	4.14.0	True	False	False	37m
machine-api	4.14.0	True	False	False	29m
machine approver	4.14.0	True	False	False	37m
machine-config	4.14.0	True	False	False	36m
marketplace	4.14.0	True	False	False	37m
monitoring	4.14.0	True	False	False	29m
network	4.14.0	True	False	False	38m
node-tuning	4.14.0	True	False	False	37m
openshift-apiserver	4.14.0	True	False	False	32m
openshift-controller-manager	4.14.0	True	False	False	30m
openshift-samples	4.14.0	True	False	False	32m
operator-lifecycle-manager	4.14.0	True	False	False	37m
operator-lifecycle-manager-catalog	4.14.0	True	False	False	37m
operator-lifecycle-manager-packageserver	4.14.0	True	False	False	32m
service-ca	4.14.0	True	False	False	38m
storage	4.14.0	True	False	False	37m

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

```
$ ./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.



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

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

- 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	
...			

- 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.

- For an installation with Fibre Channel Protocol (FCP), additional steps are required to enable multipathing. Do not enable multipathing during installation.

See "Enabling multipathing with kernel arguments on RHCOS" in the *Postinstallation machine configuration tasks* documentation for more information.

Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.14, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to [OpenShift Cluster Manager](#).

After you confirm that your [OpenShift Cluster Manager](#) inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, [use subscription watch](#) to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

Additional resources

- See [About remote health monitoring](#) for more information about the Telemetry service

Next steps

- [Validating an installation](#).
- [Customize your cluster](#).
- If necessary, you can [opt out of remote health reporting](#).
- [Set up your registry and configure registry storage](#).