OpenShift Container Platform 4.12

Installing on bare metal

Installing OpenShift Container Platform on bare metal

Last Updated: 2026-01-28

OpenShift Container Platform 4.12 Installing on bare metal

Installing OpenShift Container Platform on bare metal

Legal Notice

Copyright © Red Hat.

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document describes how to install OpenShift Container Platform on bare metal.

Table of Contents

CHAPTER 1. PREPARING FOR BARE METAL CLUSTER INSTALLATION 8
1.1. PREREQUISITES 8
1.2. BARE-METAL CLUSTER INSTALLATION REQUIREMENTS FOR OPENSHIFT VIRTUALIZATION 8
1.2.1. High availability requirements for OpenShift Virtualization 8
1.2.2. Live migration requirements for OpenShift Virtualization 8
1.2.3. SR-IOV requirements for OpenShift Virtualization 8
1.3. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON BARE METAL 9
1.3.1. Installing a cluster on installer-provisioned infrastructure 9
1.3.2. Installing a cluster on user-provisioned infrastructure 9

CHAPTER 2. INSTALLING A USER-PROVISIONED CLUSTER ON BARE METAL 11
2.1. PREREQUISITES 11
2.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 11
2.3. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 12
2.3.1. Required machines for cluster installation 12
2.3.2. Minimum resource requirements for cluster installation 13
2.3.3. Certificate signing requests management 13
2.3.4. Requirements for baremetal clusters on vSphere 14
2.3.5. Networking requirements for user-provisioned infrastructure 14
2.3.5.1. Setting the cluster node hostnames through DHCP 14
2.3.5.2. Network connectivity requirements 15
NTP configuration for user-provisioned infrastructure 16
2.3.6. User-provisioned DNS requirements 16
2.3.6.1. Example DNS configuration for user-provisioned clusters 18
2.3.7. Load balancing requirements for user-provisioned infrastructure 20
2.3.7.1. Example load balancer configuration for user-provisioned clusters 22
2.4. PREPARING THE USER-PROVISIONED INFRASTRUCTURE 24
2.5. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE 26
2.6. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 29
2.7. OBTAINING THE INSTALLATION PROGRAM 30
2.8. INSTALLING THE OPENSHIFT CLI BY DOWNLOADING THE BINARY 31
Installing the OpenShift CLI on Linux 31
Installing the OpenShift CLI on Windows 32
Installing the OpenShift CLI on macOS 32
2.9. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 33
2.9.1. Installation configuration parameters 34
2.9.1.1. Required configuration parameters 34
2.9.1.2. Network configuration parameters 36
2.9.1.3. Optional configuration parameters 38
2.9.2. Sample install-config.yaml file for bare metal 44
2.9.3. Configuring the cluster-wide proxy during installation 47
2.9.4. Configuring a three-node cluster 49
2.10. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES 50
2.11. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 52
2.11.1. Installing RHCOS by using an ISO image 53
2.11.2. Installing RHCOS by using PXE or iPXE booting 56
2.11.3. Advanced RHCOS installation configuration 61
2.11.3.1. Using advanced networking options for PXE and ISO installations 61
2.11.3.2. Disk partitioning 62
2.11.3.2.1. Creating a separate /var partition 63

2.11.3.2.2. Retaining existing partitions 65
2.11.3.3. Identifying Ignition configs 66
2.11.3.4. Default console configuration 66
2.11.3.5. Enabling the serial console for PXE and ISO installations 67
2.11.3.6. Customizing a live RHCOS ISO or PXE install 68
2.11.3.7. Customizing a live RHCOS ISO image 68
2.11.3.7.1. Modifying a live install ISO image to enable the serial console 69
2.11.3.7.2. Modifying a live install ISO image to use a custom certificate authority 70
2.11.3.7.3. Modifying a live install ISO image with customized network settings 70
2.11.3.8. Customizing a live RHCOS PXE environment 71
2.11.3.8.1. Modifying a live install PXE environment to enable the serial console 72
2.11.3.8.2. Modifying a live install PXE environment to use a custom certificate authority 73
2.11.3.8.3. Modifying a live install PXE environment with customized network settings 73
2.11.3.9. Advanced RHCOS installation reference 75
2.11.3.9.1. Networking and bonding options for ISO installations 75
Configuring DHCP or static IP addresses 75
Configuring an IP address without a static hostname 76
Specifying multiple network interfaces 76
Configuring default gateway and route 76
Disabling DHCP on a single interface 77
Combining DHCP and static IP configurations 77
Configuring VLANs on individual interfaces 77
Providing multiple DNS servers 77
Bonding multiple network interfaces to a single interface 77
Bonding multiple network interfaces to a single interface 78
Using network teaming 78
2.11.3.9.2. coreos-installer options for ISO and PXE installations 78
2.11.3.9.3. coreos.inst boot options for ISO or PXE installations 83
2.11.4. Enabling multipathing with kernel arguments on RHCOS 84
2.11.4.1. Enabling multipathing on secondary disks 86
2.12. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 88
2.13. LOGGING IN TO THE CLUSTER BY USING THE CLI 89
2.14. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 90
2.15. INITIAL OPERATOR CONFIGURATION 93
2.15.1. Image registry removed during installation 94
2.15.2. Image registry storage configuration 94
2.15.2.1. Configuring registry storage for bare metal and other manual installations 94
2.15.2.2. Configuring storage for the image registry in non-production clusters 96
2.15.2.3. Configuring block registry storage for bare metal 96
2.16. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 98
2.17. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM 100
2.18. NEXT STEPS 100

CHAPTER 3. INSTALLING A USER-PROVISIONED BARE METAL CLUSTER WITH NETWORK CUSTOMIZATIONS 101
3.1. PREREQUISITES 101
3.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 101
3.3. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 102
3.3.1. Required machines for cluster installation 102
3.3.2. Minimum resource requirements for cluster installation 102
3.3.3. Certificate signing requests management 103
3.3.4. Networking requirements for user-provisioned infrastructure 104
3.3.4.1. Setting the cluster node hostnames through DHCP 104

3.3.4.2. Network connectivity requirements 104
NTP configuration for user-provisioned infrastructure 106
3.3.5. User-provisioned DNS requirements 106
3.3.5.1. Example DNS configuration for user-provisioned clusters 108
3.3.6. Load balancing requirements for user-provisioned infrastructure 110
3.3.6.1. Example load balancer configuration for user-provisioned clusters 112
3.4. PREPARING THE USER-PROVISIONED INFRASTRUCTURE 114
3.5. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE 116
3.6. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 118
3.7. OBTAINING THE INSTALLATION PROGRAM 120
3.8. INSTALLING THE OPENSHIFT CLI BY DOWNLOADING THE BINARY 121
Installing the OpenShift CLI on Linux 121
Installing the OpenShift CLI on Windows 122
Installing the OpenShift CLI on macOS 122
3.9. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 123
3.9.1. Installation configuration parameters 124
3.9.1.1. Required configuration parameters 124
3.9.1.2. Network configuration parameters 125
3.9.1.3. Optional configuration parameters 128
3.9.2. Sample install-config.yaml file for bare metal 133
3.10. NETWORK CONFIGURATION PHASES 136
3.11. SPECIFYING ADVANCED NETWORK CONFIGURATION 137
3.12. CLUSTER NETWORK OPERATOR CONFIGURATION 138
3.12.1. Cluster Network Operator configuration object 138
defaultNetwork object configuration 139
Configuration for the OpenShift SDN network plugin 140
Configuration for the OVN-Kubernetes network plugin 141
kubeProxyConfig object configuration 146
3.13. CREATING THE IGNITION CONFIG FILES 146
3.14. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 148
3.14.1. Installing RHCOS by using an ISO image 149
3.14.2. Installing RHCOS by using PXE or iPXE booting 152
3.14.3. Advanced RHCOS installation configuration 157
3.14.3.1. Using advanced networking options for PXE and ISO installations 157
3.14.3.2. Disk partitioning 158
3.14.3.2.1. Creating a separate /var partition 159
3.14.3.2.2. Retaining existing partitions 161
3.14.3.3. Identifying Ignition configs 162
3.14.3.4. Default console configuration 162
3.14.3.5. Enabling the serial console for PXE and ISO installations 163
3.14.3.6. Customizing a live RHCOS ISO or PXE install 164
3.14.3.7. Customizing a live RHCOS ISO image 164
3.14.3.7.1. Modifying a live install ISO image to enable the serial console 165
3.14.3.7.2. Modifying a live install ISO image to use a custom certificate authority 166
3.14.3.7.3. Modifying a live install ISO image with customized network settings 166
3.14.3.8. Customizing a live RHCOS PXE environment 167
3.14.3.8.1. Modifying a live install PXE environment to enable the serial console 168
3.14.3.8.2. Modifying a live install PXE environment to use a custom certificate authority 169
3.14.3.8.3. Modifying a live install PXE environment with customized network settings 169
3.14.3.9. Advanced RHCOS installation reference 171
3.14.3.9.1. Networking and bonding options for ISO installations 171
Configuring DHCP or static IP addresses 171

Configuring an IP address without a static hostname 172
Specifying multiple network interfaces 172
Configuring default gateway and route 172
Disabling DHCP on a single interface 173
Combining DHCP and static IP configurations 173
Configuring VLANs on individual interfaces 173
Providing multiple DNS servers 173
Bonding multiple network interfaces to a single interface 173
Bonding multiple network interfaces to a single interface 174
Using network teaming 174
3.14.3.9.2. coreos-installer options for ISO and PXE installations 174
3.14.3.9.3. coreos.inst boot options for ISO or PXE installations 179
3.14.4. Enabling multipathing with kernel arguments on RHCOS 180
3.14.4.1. Enabling multipathing on secondary disks 182
3.15. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 184
3.16. LOGGING IN TO THE CLUSTER BY USING THE CLI 185
3.17. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 186
3.18. INITIAL OPERATOR CONFIGURATION 188
3.18.1. Image registry removed during installation 190
3.18.2. Image registry storage configuration 190
3.18.3. Configuring block registry storage for bare metal 190
3.19. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 191
3.20. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM 194
3.21. NEXT STEPS 194

CHAPTER 4. INSTALLING A USER-PROVISIONED BARE METAL CLUSTER ON A RESTRICTED NETWORK . . 195
4.1. PREREQUISITES 195
4.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS 195
4.2.1. Additional limits 196
4.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 196
4.4. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 196
4.4.1. Required machines for cluster installation 197
4.4.2. Minimum resource requirements for cluster installation 197
4.4.3. Certificate signing requests management 198
4.4.4. Networking requirements for user-provisioned infrastructure 198
4.4.4.1. Setting the cluster node hostnames through DHCP 199
4.4.4.2. Network connectivity requirements 199
NTP configuration for user-provisioned infrastructure 200
4.4.5. User-provisioned DNS requirements 201
4.4.5.1. Example DNS configuration for user-provisioned clusters 203
4.4.6. Load balancing requirements for user-provisioned infrastructure 205
4.4.6.1. Example load balancer configuration for user-provisioned clusters 207
4.5. PREPARING THE USER-PROVISIONED INFRASTRUCTURE 209
4.6. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE 211
4.7. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 213
4.8. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 215
4.8.1. Installation configuration parameters 216
4.8.1.1. Required configuration parameters 216
4.8.1.2. Network configuration parameters 218
4.8.1.3. Optional configuration parameters 220
4.8.2. Sample install-config.yaml file for bare metal 226
4.8.3. Configuring the cluster-wide proxy during installation 229

4.8.4. Configuring a three-node cluster 231
4.9. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES 232
4.10. CONFIGURING CHRONY TIME SERVICE 234
4.11. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 235
4.11.1. Installing RHCOS by using an ISO image 236
4.11.2. Installing RHCOS by using PXE or iPXE booting 239
4.11.3. Advanced RHCOS installation configuration 244
4.11.3.1. Using advanced networking options for PXE and ISO installations 244
4.11.3.2. Disk partitioning 245
4.11.3.2.1. Creating a separate /var partition 246
4.11.3.2.2. Retaining existing partitions 248
4.11.3.3. Identifying Ignition configs 249
4.11.3.4. Default console configuration 250
4.11.3.5. Enabling the serial console for PXE and ISO installations 251
4.11.3.6. Customizing a live RHCOS ISO or PXE install 251
4.11.3.7. Customizing a live RHCOS ISO image 252
4.11.3.7.1. Modifying a live install ISO image to enable the serial console 252
4.11.3.7.2. Modifying a live install ISO image to use a custom certificate authority 253
4.11.3.7.3. Modifying a live install ISO image with customized network settings 254
4.11.3.8. Customizing a live RHCOS PXE environment 255
4.11.3.8.1. Modifying a live install PXE environment to enable the serial console 256
4.11.3.8.2. Modifying a live install PXE environment to use a custom certificate authority 256
4.11.3.8.3. Modifying a live install PXE environment with customized network settings 257
4.11.3.9. Advanced RHCOS installation reference 258
4.11.3.9.1. Networking and bonding options for ISO installations 258
Configuring DHCP or static IP addresses 259
Configuring an IP address without a static hostname 259
Specifying multiple network interfaces 260
Configuring default gateway and route 260
Disabling DHCP on a single interface 260
Combining DHCP and static IP configurations 260
Configuring VLANs on individual interfaces 261
Providing multiple DNS servers 261
Bonding multiple network interfaces to a single interface 261
Bonding multiple network interfaces to a single interface 261
Using network teaming 262
4.11.3.9.2. coreos-installer options for ISO and PXE installations 262
4.11.3.9.3. coreos.inst boot options for ISO or PXE installations 266
4.11.4. Enabling multipathing with kernel arguments on RHCOS 268
4.11.4.1. Enabling multipathing on secondary disks 270
4.12. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 271
4.13. LOGGING IN TO THE CLUSTER BY USING THE CLI 272
4.14. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 273
4.15. INITIAL OPERATOR CONFIGURATION 276
4.15.1. Disabling the default OperatorHub catalog sources 277
4.15.2. Image registry storage configuration 277
4.15.2.1. Changing the image registry's management state 278
4.15.2.2. Configuring registry storage for bare metal and other manual installations 278
4.15.2.3. Configuring storage for the image registry in non-production clusters 279
4.15.2.4. Configuring block registry storage for bare metal 280
4.16. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 281
4.17. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM 284

4.18. NEXT STEPS 284

CHAPTER 1. PREPARING FOR BARE METAL CLUSTER INSTALLATION

1.1. PREREQUISITES

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You have read the documentation on selecting a cluster installation method and preparing it for users.

1.2. BARE-METAL CLUSTER INSTALLATION REQUIREMENTS FOR OPENSHIFT VIRTUALIZATION

If you plan to use OpenShift Virtualization on a bare-metal cluster, you must ensure that your cluster is configured correctly during installation. This is because OpenShift Virtualization requires certain settings that cannot be changed after a cluster is installed.

1.2.1. High availability requirements for OpenShift Virtualization

When discussing high availability (HA) features in the context of OpenShift Virtualization, this refers only to the replication model of the core cluster components, determined by the **controlPlaneTopology** and **infrastructureTopology** fields in the **Infrastructure** custom resource (CR). Setting these fields to **HighlyAvailable** offers component redundancy, which is distinct from general cluster-wide application HA. Setting these fields to **SingleReplica** disables component redundancy, and therefore disables OpenShift Virtualization HA features.

If you plan to use OpenShift Virtualization HA features, you must have three control plane nodes at the time of cluster installation. The **controlPlaneTopology** status in the **Infrastructure** CR for the cluster must be **HighlyAvailable**.

NOTE

You can install OpenShift Virtualization on a single-node cluster, but single-node OpenShift does not support HA features.

1.2.2. Live migration requirements for OpenShift Virtualization

- If you plan to use live migration, you must have multiple worker nodes. The **infrastructureTopology** status in the **Infrastructure** CR for the cluster must be **HighlyAvailable**, and a minimum of three worker nodes is recommended.

  NOTE

  You can install OpenShift Virtualization on a single-node cluster, but single-node OpenShift does not support live migration.

- Live migration requires shared storage. Storage for OpenShift Virtualization must support and use the ReadWriteMany (RWX) access mode.

1.2.3. SR-IOV requirements for OpenShift Virtualization

If you plan to use Single Root I/O Virtualization (SR-IOV), ensure that your network interface controllers (NICs) are supported by OpenShift Container Platform.

Additional resources

- Preparing your cluster for OpenShift Virtualization
- Getting started with OpenShift Virtualization
- About Single Root I/O Virtualization (SR-IOV) hardware networks
- Connecting a virtual machine to an SR-IOV network

1.3. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON BARE METAL

The OpenShift Container Platform installation program offers four methods for deploying a cluster:

- **Interactive**: You can deploy a cluster with the web-based Assisted Installer. This is the recommended approach for clusters with networks connected to the internet. The Assisted Installer is the easiest way to install OpenShift Container Platform, it provides smart defaults, and it performs pre-flight validations before installing the cluster. It also provides a RESTful API for automation and advanced configuration scenarios.
- **Local Agent-based**: You can deploy a cluster locally with the agent-based installer for air-gapped or restricted networks. It provides many of the benefits of the Assisted Installer, but you must download and configure the agent-based installer first. Configuration is done with a commandline interface. This approach is ideal for air-gapped or restricted networks.
- **Automated**: You can deploy a cluster on installer-provisioned infrastructure and the cluster it maintains. The installer uses each cluster host's baseboard management controller (BMC) for provisioning. You can deploy clusters with both connected or air-gapped or restricted networks.
- **Full control**: You can deploy a cluster on infrastructure that you prepare and maintain , which provides maximum customizability. You can deploy clusters with both connected or air-gapped or restricted networks.

The clusters have the following characteristics:

- Highly available infrastructure with no single points of failure is available by default.
- Administrators maintain control over what updates are applied and when.

See Installation process for more information about installer-provisioned and user-provisioned installation processes.

1.3.1. Installing a cluster on installer-provisioned infrastructure

You can install a cluster on bare metal infrastructure that is provisioned by the OpenShift Container Platform installation program, by using the following method:

- **Installing an installer-provisioned cluster on bare metal**: You can install OpenShift Container Platform on bare metal by using installer provisioning.

1.3.2. Installing a cluster on user-provisioned infrastructure

You can install a cluster on bare metal infrastructure that you provision, by using one of the following methods:

- **Installing a user-provisioned cluster on bare metal**: You can install OpenShift Container Platform on bare metal infrastructure that you provision. For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.
- **Installing a user-provisioned bare metal cluster with network customizations**: You can install a bare metal cluster on user-provisioned infrastructure with network-customizations. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations. Most of the network customizations must be applied at the installation stage.
- **Installing a user-provisioned bare metal cluster on a restricted network**: You can install a user-provisioned bare metal cluster on a restricted or disconnected network by using a mirror registry. You can also use this installation method to ensure that your clusters only use container images that satisfy your organizational controls on external content.

CHAPTER 2. INSTALLING A USER-PROVISIONED CLUSTER ON BARE METAL

In OpenShift Container Platform 4.12, you can install a cluster on bare metal infrastructure that you provision.

IMPORTANT

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

2.1. PREREQUISITES

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

NOTE

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

2.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.12, you require access to the internet to install your cluster.

You must have internet access to:

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

IMPORTANT

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

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.

2.3. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

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

2.3.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 2.1. 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.

NOTE

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.

IMPORTANT

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

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

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

2.3.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 2.2. 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

1. 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.

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

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

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

2.3.3. 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.

2.3.4. 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.

2.3.5. 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.

NOTE

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.

2.3.5.1. Setting the cluster node hostnames through DHCP

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

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

2.3.5.2. Network connectivity requirements

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

This section provides details about the ports that are required.

IMPORTANT

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

Table 2.3. 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
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
	123	Network Time Protocol (NTP) on UDP port **123** If an external NTP time server is configured, you must open UDP port **123**.
TCP/UDP	**30000-32767**	Kubernetes node port

Protocol	Port	Description
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 2.4. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 2.5. 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

2.3.6. 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.

NOTE

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

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

Table 2.6. Required DNS records

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

Component	Record	Description
Control plane machines	**<control_plane><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the control plane nodes. These records must be resolvable by the nodes within the cluster.
Compute machines	**<compute><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA or CNAME records and DNS PTR records to identify each machine for the worker nodes. These records must be resolvable by the nodes within the cluster.

NOTE

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

TIP

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

2.3.6.1. 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.

Example 2.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.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 (6)
control-plane2.ocp4.example.com. IN A 192.168.1.99 (7)
;
compute0.ocp4.example.com. IN A 192.168.1.11 (8)
compute1.ocp4.example.com. IN A 192.168.1.7 (9)
;
;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.
3. 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.

   NOTE

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

4. Provides name resolution for the bootstrap machine.

5 6 7 Provides name resolution for the control plane machines.

8 9 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.

Example 2.2. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
```

```
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
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. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;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. 5. 6. Provides reverse DNS resolution for the control plane machines.
7. 8. Provides reverse DNS resolution for the compute machines.

NOTE

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

Additional resources

- Validating DNS resolution for user-provisioned infrastructure

2.3.7. Load balancing requirements for user-provisioned infrastructure

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

NOTE

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

The load balancing infrastructure must meet the following requirements:

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

   - Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.
   - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

   IMPORTANT

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

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

   Table 2.7. 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 **/readyz** endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	X		Machine config server

   NOTE

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

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

   - Layer 4 load balancing only. This can be referred to as Raw TCP or SSL Passthrough mode.

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

TIP

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

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

Table 2.8. 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

NOTE

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

2.3.7.1. 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.

NOTE

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

Example 2.3. 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 (4)
  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 (5)
  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 (6)
  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 4 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.

5 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.

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

NOTE

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

TIP

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

2.4. 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

1. If you are using DHCP to provide the IP networking configuration to your cluster nodes, configure your DHCP service.
   a. Add persistent IP addresses for the nodes to your DHCP server configuration. In your configuration, match the MAC address of the relevant network interface to the intended IP address for each node.

b. When you use DHCP to configure IP addressing for the cluster machines, the machines also obtain the DNS server information through DHCP. Define the persistent DNS server address that is used by the cluster nodes through your DHCP server configuration.

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

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

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

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

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

> **IMPORTANT**
>
> By default, port **1936** is accessible for an OpenShift Container Platform cluster, because each control plane node needs access to this port.
>
> Avoid using the Ingress load balancer to expose this port, because doing so might result in the exposure of sensitive information, such as statistics and metrics, related to Ingress Controllers.

4. Setup the required DNS infrastructure for your cluster.

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

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

5. Validate your DNS configuration.

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

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

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

NOTE

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

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

2.5. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE

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

IMPORTANT

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

Prerequisites

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

Procedure

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

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

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

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

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

Example output

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

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

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

Example output

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

NOTE

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

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

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

Example output

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

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

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

Example output

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

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

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

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

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

Example output

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

NOTE

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

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

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

Example output

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

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

Additional resources

- User-provisioned DNS requirements
- Load balancing requirements for user-provisioned infrastructure

2.6. 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.

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

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

Procedure

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

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

(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.

> **NOTE**
>
> If you plan to install an OpenShift Container Platform cluster that uses FIPS validated or Modules In Process cryptographic libraries on the **x86_64**, **ppc64le**, and **s390x** architectures. do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

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

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

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

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

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

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

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

   Example output

```
Agent pid 31874
```

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

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

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

2.7. OBTAINING THE INSTALLATION PROGRAM

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

Prerequisites

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

Procedure

1. 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.

2. Select your infrastructure provider.

3. 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.

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

> **IMPORTANT**
>
> Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

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

5. Download your installation pull secret from the 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.

2.8. 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.

> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.12. 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

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.12 Linux Client** entry and save the file.
5. Unpack the archive:

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

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

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

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

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.12 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

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

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.12 macOS Client** entry and save the file.

> **NOTE**
>
> For macOS arm64, choose the **OpenShift v4.12 macOS arm64 Client** entry.

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

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

2.9. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE

Installing the cluster requires that you manually create the installation configuration file.

Prerequisites

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

Procedure

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

```
$ mkdir <installation_directory>
```

IMPORTANT

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

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

NOTE

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

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

IMPORTANT

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

2.9.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide a customized **install-config.yaml** installation configuration file that describes the details for your environment.

NOTE

After installation, you cannot modify these parameters in the **install-config.yaml** file.

2.9.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 2.9. Required parameters

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installation program may also support older API versions.	String

Parameter	Description	Values
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the **baseDomain** and **metadata.name** parameter values that uses the **<metadata.name>.<baseDomain>** format.	A fully-qualified domain or subdomain name, such as **example.com**.
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.	String of lowercase letters and hyphens (**-**), such as **dev**.
platform	The configuration for the specific platform upon which to perform the installation: **alibabacloud**, **aws**, **baremetal**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**. For additional information about **platform.<platform>** parameters, consult the table for your specific platform that follows.	Object

Parameter	Description	Values
pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

2.9.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

- If you use the Red Hat OpenShift Networking OVN-Kubernetes network plugin, both IPv4 and IPv6 address families are supported.
- If you use the Red Hat OpenShift Networking OpenShift SDN network plugin, only the IPv4 address family is supported.

If you configure your cluster to use both IP address families, review the following requirements:

- Both IP families must use the same network interface for the default gateway.
- Both IP families must have the default gateway.
- You must specify IPv4 and IPv6 addresses in the same order for all network configuration parameters. For example, in the following configuration IPv4 addresses are listed before IPv6 addresses.

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```

NOTE

Globalnet is not supported with Red Hat OpenShift Data Foundation disaster recovery solutions. For regional disaster recovery scenarios, ensure that you use a nonoverlapping range of private IP addresses for the cluster and service networks in each cluster.

Table 2.10. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object **NOTE** You cannot modify parameters specified by the **networking** object after installation.
networking.networkType	The Red Hat OpenShift Networking network plugin to install.	Either **OpenShiftSDN** or **OVNKubernetes**. **OpenShiftSDN** is a CNI plugin for all-Linux networks. **OVNKubernetes** is a CNI plugin for Linux networks and hybrid networks that contain both Linux and Windows servers. The default value is **OVNKubernetes**.

networking.clusterNetwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		  - cidr: fd01::/48
		    hostPrefix: 64
		``

networking.clusterNetwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. If you use the OpenShift SDN network plugin, specify an IPv4 network. If you use the OVN-Kubernetes network plugin, you can specify IPv4 and IPv6 networks.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**. The prefix length for an IPv6 block is between **0** and **128**. For example, **10.128.0.0/14** or **fd01::/48**.
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a **/23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 - 23) - 2) pod IP addresses.	A subnet prefix. For an IPv4 network the default value is **23**. For an IPv6 network the default value is **64**. The default value is also the minimum value for IPv6.

Parameter	Description	Values

networking.serviceNetwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network plugins support only a single IP address block for the service network.
	
	If you use the OVN-Kubernetes network plugin, you can specify an IP address block for both of the IPv4 and IPv6 address families.
		An array with an IP address block in CIDR format. For example:
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`
		`  - fd02::/112`

networking.machineNetwork	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16** or **fd00::/48**. **NOTE** Set the **networking.machineNetwork** to match the CIDR that the preferred NIC resides in.

2.9.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 2.11. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String

Parameter	Description	Values
capabilities	Controls the installation of optional core cluster components. You can reduce the footprint of your OpenShift Container Platform cluster by disabling optional components. For more information, see the "Cluster capabilities" page in *Installing*.	String array
capabilities.baselineCapabilitySet	Selects an initial set of optional capabilities to enable. Valid values are **None**, **v4.11**, **v4.12** and **vCurrent**. The default value is **vCurrent**.	String
capabilities.additionalEnabledCapabilities	Extends the set of optional capabilities beyond what you specify in **baselineCapabilitySet**. You may specify multiple capabilities in this parameter.	String array
compute	The configuration for the machines that comprise the compute nodes.	Array of **MachinePool** objects.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**

Parameter	Description	Values
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**alibabacloud**, **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to **2**. The default value is **3**.
featureSet	Enables the cluster for a feature set. A feature set is a collection of OpenShift Container Platform features that are not enabled by default. For more information about enabling a feature set during installation, see "Enabling features using feature gates".	String. The name of the feature set to enable, such as **TechPreviewNoUpgrade**.
controlPlane	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.	String

Parameter	Description	Values
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platform	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**alibabacloud**, **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.

Parameter	Description	Values
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. NOTE Not all CCO modes are supported for all cloud providers. For more information about CCO modes, see the *Cloud Credential Operator* entry in the *Cluster Operators reference* content. NOTE If your AWS account has service control policies (SCP) enabled, you must configure the **credentialsMode** parameter to **Mint**, **Passthrough** or **Manual**.	**Mint**, **Passthrough**, **Manual** or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead. **IMPORTANT** To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64**, **ppc64le**, and **s390x** architectures. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. The default value is **External**. Setting this field to **Internal** is not supported on non-cloud platforms. IMPORTANT If the value of the field is set to **Internal**, the cluster will become non-functional. For more information, refer to BZ#1953035.
sshKey	The SSH key to authenticate access to your cluster machines. NOTE For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.	For example, **sshKey: ssh-ed25519 AAAA..**.

2.9.2. 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: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths": ...}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
```

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

2, 5. The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

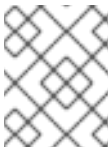
3, 6. Specifies whether to enable or disable simultaneous multithreading (SMT), or hyperthreading. By default, SMT is enabled to increase the performance of the cores in your machines. You can disable it by setting the parameter value to **Disabled**. If you disable SMT, you must disable it in all cluster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

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

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

4. 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.



NOTE

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

7. 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.

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

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

NOTE

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

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

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

12. 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.

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

IMPORTANT

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

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

IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64**, **ppc64le**, and **s390x** architectures.

15 The pull secret from the 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.

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

NOTE

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

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 4.12 for more information about the features provided by each capability.

2.9.3. Configuring the cluster-wide proxy during installation

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

NOTE

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

Prerequisites

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

NOTE

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

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

Procedure

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

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> (1)
  httpsProxy: 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.

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

(4) If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **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.

(5) 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**.

NOTE

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

NOTE

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

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

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

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

NOTE

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

2.9.4. 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
```

NOTE

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

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

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

2.10. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES

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

IMPORTANT

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

Prerequisites

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

Procedure

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

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

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

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

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

2. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:
   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.
   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.
   c. Save and exit the file.
3. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

2.11. 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.

NOTE

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

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

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

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

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

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

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

2.11.1. 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

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

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

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

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

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

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

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

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

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

Example output

```
"location": "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

IMPORTANT

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

ISO file names resemble the following example:

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

5. Use the ISO to start the RHCOS installation. Use one of the following installation options:
   - Burn the ISO image to a disk and boot it directly.
   - Use ISO redirection by using a lights-out management (LOM) interface.

6. Boot the RHCOS ISO image without specifying any options or interrupting the live boot sequence. Wait for the installer to boot into a shell prompt in the RHCOS live environment.

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

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

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

(1) (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.

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

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

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

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

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

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

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

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

Example command

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

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

IMPORTANT

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

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

NOTE

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

2.11.2. Installing RHCOS by using PXE or iPXE booting

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

Prerequisites

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

Procedure

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

IMPORTANT

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

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

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

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

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

Example output

```
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-
kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-
initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-
rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/49.84.202110081256-0/ppc64le/rhcos-
<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-
initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-
rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-kernel-
s390x"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-
initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-
rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-kernel-
x86_64"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-
```

```
initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-
rootfs.x86_64.img"
```

IMPORTANT

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

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

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

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

IMPORTANT

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

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.
6. Configure PXE or iPXE installation for the RHCOS images and begin the installation. 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_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (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**

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

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

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

1. 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.
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 location of the **initramfs** file that you uploaded to your HTTP server.

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

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

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

```
menuentry 'Install CoreOS' {
  linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
  initrd rhcos-<version>-live-initramfs.<architecture>.img (3)
}
```

(1) 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.

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

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

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

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

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

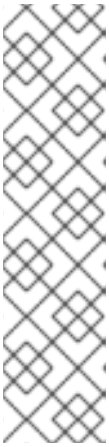
Example command

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

10. Continue to create the machines for your cluster.

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

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



NOTE

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

2.11.3. 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.

2.11.3.1. Using advanced networking options for PXE and ISO installations

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

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

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

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

To configure an ISO installation, use the following procedure.

Procedure

1. Boot the ISO installer.
2. From the live system shell prompt, configure networking for the live system using available RHEL tools, such as **nmcli** or **nmtui**.

3. Run the **coreos-installer** command to install the system, adding the **--copy-network** option to copy networking configuration. For example:

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

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

4. 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.

2.11.3.2. Disk partitioning

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 you override the default partitioning configuration. During the RHCOS installation, the size of the root file system is increased to use any remaining available space on the target device.

> **IMPORTANT**
>
> The use of a custom partition scheme on your node might result in OpenShift Container Platform not monitoring or alerting on some node partitions. If you override the default partitioning, see Understanding OpenShift File System Monitoring (eviction conditions) for more information about how OpenShift Container Platform monitors your host file systems.

OpenShift Container Platform monitors the following two filesystem identifiers:

- **nodefs**, which is the filesystem that contains **/var/lib/kubelet**
- **imagefs**, which is the filesystem that contains **/var/lib/containers**

For the default partition scheme, **nodefs** and **imagefs** monitor the same root filesystem, /.

To override the default partitioning when installing RHCOS on an OpenShift Container Platform cluster node, you must create separate partitions.

> **IMPORTANT**
>
> 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.

Consider a situation where you want to add a separate storage partition for your containers and container images. For example, by mounting **/var/lib/containers** in a separate partition, the kubelet separately monitors **/var/lib/containers** as the **imagefs** directory and the root file system as the **nodefs** directory.

> **IMPORTANT**
>
> If you have resized your disk size to host a larger file system, consider creating a separate **/var/lib/containers** partition. Consider resizing a disk that has an **xfs** format to reduce CPU time issues caused by a high number of allocation groups.

2.11.3.2.1. 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.

> **IMPORTANT**
>
> 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

1. 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>
```

2. 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: 4.12.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> (1)
    partitions:
    - label: var
      start_mib: <partition_start_offset> (2)
      size_mib: <partition_size> (3)
      number: 5
  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.
2. 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 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.

NOTE

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.

3. 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
```

4. 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.

2.11.3.2.2. 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.

> **NOTE**
>
> 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/sda
```

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

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

This example preserves partitions 5 and higher:

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

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

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
```

2.11.3.3. Identifying Ignition configs

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

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

  > **IMPORTANT**
  >
  > It is not recommended to modify these 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.

2.11.3.4. Default console configuration

Red Hat Enterprise Linux CoreOS (RHCOS) nodes installed from an OpenShift Container Platform 4.12 boot image use a default console that is meant to accomodate 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.

NOTE

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

2.11.3.5. 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

1. Boot the ISO installer.
2. 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/sda
```

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

(2) 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 **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.

3. Reboot into the installed system.

> **NOTE**
>
> 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.

2.11.3.6. 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.

2.11.3.7. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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/sda (2)
```

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

(2) When you specify this option, the ISO image automatically runs an installation. Otherwise, the image remains configured for installation, but does not install automatically unless you specify the **coreos.inst.install_dev** kernel argument.

3. 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.

2.11.3.7.1. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda (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. In this case, **/dev/sda**. 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.

> **NOTE**
>
> 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.

3. 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.

2.11.3.7.2. 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.

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

Procedure

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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
```

> **IMPORTANT**
>
> 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.

2.11.3.7.3. 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.

> **WARNING**
>
> 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto
```

3. 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
slave-type=bond
```

4. 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
slave-type=bond
```

5. 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.

2.11.3.8. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda \ (2)
  -o rhcos-<version>-custom-initramfs.x86_64.img (3)
```

(1) The Ignition config file that is generated from **openshift-installer**.

(2) When you specify this option, the PXE environment automatically runs an install. Otherwise, the image remains 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.

2.11.3.8.1. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda \ (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.

2.11.3.8.2. 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.

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

Procedure

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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
```

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.

> **IMPORTANT**
>
> 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.

2.11.3.8.3. 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.

> **WARNING**
>
> 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto
```

3. 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
slave-type=bond
```

4. 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
slave-type=bond
```

5. 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
```

6. 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.

2.11.3.9. 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.

2.11.3.9.1. 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.

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

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

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

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

NOTE

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

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.

NOTE

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

- 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 network interfaces to a single interface
Optional: You can configure VLANs on bonded interfaces by using the **vlan=** parameter and to use DHCP, for example:

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

Use the following example to configure the bonded interface with a VLAN and to use a static IP address:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

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**).

NOTE

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

Use the following example to configure a network team:

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

2.11.3.9.2. **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 2.12. **coreos-installer** subcommands, command-line options, and arguments

coreos-installer install subcommand	
Subcommand	*Description*
$ coreos-installer install <options> <device>	Embed an Ignition config in an ISO image.
coreos-installer install subcommand options	
Option	*Description*
-u, **--image-url <url>**	Specify the image URL manually.
-f, **--image-file <path>**	Specify a local image file manually. Used for debugging.
-i, --ignition-file <path>	Embed an Ignition config from a file.
-I, **--ignition-url <URL>**	Embed an Ignition config from a URL.
--ignition-hash <digest>	Digest **type-value** of the Ignition config.
-p, **--platform <name>**	Override the Ignition platform ID for the installed system.
--console <spec>	Set the kernel and bootloader console for the installed system. For more information about the format of **<spec>**, see the Linux kernel serial console documentation.
--append-karg <arg>...	Append a default kernel argument to the installed system.
--delete-karg <arg>...	Delete a default kernel argument from the installed system.

-n, **--copy-network**	Copy the network configuration from the install environment. IMPORTANT The **--copy-network** option only copies networking configuration found under **/etc/NetworkManager/system-connections**. In particular, it does not copy the system hostname.
--network-dir <path>	For use with **-n**. Default is **/etc/NetworkManager/system-connections/**.
--save-partlabel <lx>..	Save partitions with this label glob.
--save-partindex <id>…	Save partitions with this number or range.
--insecure	Skip RHCOS image signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Valid values are **x86_64** and **aarch64**.
--preserve-on-error	Do not clear partition table on error.
-h, **--help**	Print help information.
coreos-installer install subcommand argument	
Argument	*Description*
<device>	The destination device.
coreos-installer ISO subcommands	
Subcommand	*Description*
$ coreos-installer iso customize <options> <ISO_image>	Customize a RHCOS live ISO image.

coreos-installer iso reset <options> <ISO_image>	Restore a RHCOS live ISO image to default settings.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.
coreos-installer ISO customize subcommand options	
Option	*Description*
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and 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	
Subcommand	*Description*
Note that not all of these options are accepted by all subcommands.	
coreos-installer pxe customize <options> <path>	Customize a RHCOS live PXE boot config.
coreos-installer pxe ignition wrap <options>	Wrap an Ignition config in an image.
coreos-installer pxe ignition unwrap <options> <image_name>	Show the wrapped Ignition config in an image.
coreos-installer PXE customize subcommand options	
Option	*Description*
Note that not all of these options are accepted by all subcommands.	
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and 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.
post-install <path>	Run the specified script after installation.
--installer-config <path>	Apply the specified installer configuration file.

--live-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
-o, --output <path>	Write the initramfs to a new output file. **NOTE** This option is required for PXE environments.
-h, --help	Print help information.

2.11.3.9.3. **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 2.13. **coreos.inst** boot options

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to. It is recommended to use the full path, such as **/dev/sda**, although **sda** is allowed.
coreos.inst.ignition_url	Optional: The URL of the Ignition config to embed into the installed system. If no URL is specified, no Ignition config is embedded. Only HTTP and HTTPS protocols are supported.
coreos.inst.save_partlabel	Optional: Comma-separated labels of partitions to preserve during the install. Glob-style wildcards are permitted. The specified partitions do not need to exist.
coreos.inst.save_partindex	Optional: Comma-separated indexes of partitions to preserve during the install. Ranges **m-n** are permitted, and either **m** or **n** can be omitted. The specified partitions do not need to exist.

Argument	Description
coreos.inst.insecure	Optional: Permits the OS image that is specified by **coreos.inst.image_url** to be unsigned.
coreos.inst.image_url	Optional: Download and install the specified RHCOS image. • This argument should not be used in production environments and is intended for debugging purposes only. • While this argument can be used to install a version of RHCOS that does not match the live media, it is recommended that you instead use the media that matches the version you want to install. • If you are using **coreos.inst.image_url**, you must also use **coreos.inst.insecure**. This is because the bare-metal media are not GPG-signed for OpenShift Container Platform. • Only HTTP and HTTPS protocols are supported.
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. After the install finishes, you will receive a prompt that allows you to inspect what is happening during installation. This argument should not be used in production environments and is intended for debugging purposes only.
coreos.inst.platform_id	Optional: The Ignition platform ID of the platform the RHCOS image is being installed on. Default is **metal**. This option determines whether or not to request an Ignition config from the cloud provider, such as VMware. For example: **coreos.inst.platform_id=vmware**.
ignition.config.url	Optional: The URL of the Ignition config for the live boot. For example, this can be used to customize how **coreos-installer** is invoked, or to run code before or after the installation. This is different from **coreos.inst.ignition_url**, which is the Ignition config for the installed system.

2.11.4. 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.

IMPORTANT

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.

NOTE

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

Prerequisites

- You have created the Ignition config files for your cluster.
- You have reviewed *Installing RHCOS and starting the OpenShift Container Platform bootstrap process*.

Procedure

1. To enable multipath and start the **multipathd** daemon, run the following command on the installation host:

```
$ 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.

2. 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)
--ignition-url=http://host/worker.ign \
--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)
--ignition-url=http://host/worker.ign \
--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".

3. Reboot into the installed system.

4. 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.

2.11.4.1. Enabling multipathing on secondary disks

RHCOS also supports multipathing on a secondary disk. Instead of kernel arguments, you use Ignition to enable multipathing for the secondary disk at installation time.

Prerequisites

- You have read the section *Disk partitioning*.
- You have read *Enabling multipathing with kernel arguments on RHCOS* .
- You have installed the Butane utility.

Procedure

1. Create a Butane config with information similar to the following:

Example multipath-config.bu

```
variant: openshift
version: 4.12.0
systemd:
  units:
    - name: mpath-configure.service
      enabled: true
      contents: |
        [Unit]
        Description=Configure Multipath on Secondary Disk
        ConditionFirstBoot=true
        ConditionPathExists=!/etc/multipath.conf
        Before=multipathd.service (1)
        DefaultDependencies=no

        [Service]
        Type=oneshot
        ExecStart=/usr/sbin/mpathconf --enable (2)

        [Install]
        WantedBy=multi-user.target
    - name: mpath-var-lib-container.service
      enabled: true
      contents: |
        [Unit]
        Description=Set Up Multipath On /var/lib/containers
        ConditionFirstBoot=true (3)
        Requires=dev-mapper-mpatha.device
        After=dev-mapper-mpatha.device
        After=ostree-remount.service
        Before=kubelet.service
        DefaultDependencies=no

        [Service] (4)
        Type=oneshot
        ExecStart=/usr/sbin/mkfs.xfs -L containers -m reflink=1 /dev/mapper/mpatha
        ExecStart=/usr/bin/mkdir -p /var/lib/containers

        [Install]
        WantedBy=multi-user.target
    - name: var-lib-containers.mount
      enabled: true
      contents: |
        [Unit]
        Description=Mount /var/lib/containers
        After=mpath-var-lib-containers.service
        Before=kubelet.service (5)

        [Mount] (6)
        What=/dev/disk/by-label/dm-mpath-containers
        Where=/var/lib/containers
        Type=xfs
```

```
[Install]
WantedBy=multi-user.target
```

1. The configuration must be set before launching the multipath daemon.
2. Starts the **mpathconf** utility.
3. This field must be set to the value **true**.
4. Creates the filesystem and directory **/var/lib/containers**.
5. The device must be mounted before starting any nodes.
6. Mounts the device to the **/var/lib/containers** mount point. This location cannot be a symlink.

2. Create the Ignition configuration by running the following command:

```
$ butane --pretty --strict multipath-config.bu > multipath-config.ign
```

3. Continue with the rest of the first boot RHCOS installation process.

IMPORTANT

Do not add the **rd.multipath** or **root** kernel arguments on the command-line during installation unless the primary disk is also multipathed.

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.

2.12. 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

1. Monitor the bootstrap process:

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

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

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

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.25.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

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

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

IMPORTANT

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

Additional resources

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

2.13. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

2.14. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

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

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE  VERSION
master-0  Ready    master   63m  v1.25.0
master-1  Ready    master   63m  v1.25.0
master-2  Ready    master   64m  v1.25.0
```

The output lists all of the machines that you created.

NOTE

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

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

```
$ oc get csr
```

Example output

```
NAME        AGE    REQUESTOR                                          CONDITION
```

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

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

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

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

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

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

```
$ oc adm certificate approve <csr_name> (1)
```

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

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

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

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

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

```
$ oc get csr
```

Example output

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

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

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

```
$ oc adm certificate approve <csr_name> (1)
```

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

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

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

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

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.25.0
master-1  Ready     master  73m  v1.25.0
master-2  Ready     master  74m  v1.25.0
worker-0  Ready     worker  11m  v1.25.0
worker-1  Ready     worker  11m  v1.25.0
```

NOTE

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

Additional information

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

2.15. INITIAL OPERATOR CONFIGURATION

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

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False      19m
baremetal                                  4.12.0    True        False         False      37m
cloud-credential                           4.12.0    True        False         False      40m
cluster-autoscaler                         4.12.0    True        False         False      37m
config-operator                            4.12.0    True        False         False      38m
console                                    4.12.0    True        False         False      26m
csi-snapshot-controller                    4.12.0    True        False         False      37m
dns                                        4.12.0    True        False         False      37m
etcd                                       4.12.0    True        False         False      36m
image-registry                             4.12.0    True        False         False      31m
ingress                                    4.12.0    True        False         False      30m
insights                                   4.12.0    True        False         False      31m
kube-apiserver                             4.12.0    True        False         False      26m
kube-controller-manager                    4.12.0    True        False         False      36m
kube-scheduler                             4.12.0    True        False         False      36m
kube-storage-version-migrator              4.12.0    True        False         False      37m
machine-api                                4.12.0    True        False         False      29m
machine-approver                           4.12.0    True        False         False      37m
machine-config                             4.12.0    True        False         False      36m
marketplace                                4.12.0    True        False         False      37m
monitoring                                 4.12.0    True        False         False      29m
network                                    4.12.0    True        False         False      38m
node-tuning                                4.12.0    True        False         False      37m
openshift-apiserver                        4.12.0    True        False         False      32m
openshift-controller-manager               4.12.0    True        False         False      30m
openshift-samples                          4.12.0    True        False         False      32m
operator-lifecycle-manager                 4.12.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False      32m
service-ca                                 4.12.0    True        False         False      38m
storage                                    4.12.0    True        False         False      37m
```

2. 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.

2.15.1. Image registry removed during installation

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

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

2.15.2. Image registry storage configuration

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

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

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

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

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

Prerequisites

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

  IMPORTANT

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

- Must have 100Gi capacity.

Procedure

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

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

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

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

Example output

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

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

3. Check the registry configuration:

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

Example output

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

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

4. Check the **clusteroperator** status:

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

Example output

```
NAME            VERSION          AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.12             True        False         False      6h50m
```

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

- Run:

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

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

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

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

Procedure

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

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

> **WARNING**
>
> Configure this option for only non-production clusters.

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

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

Wait a few minutes and run the command again.

2.15.2.3. Configuring block registry storage for bare metal

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

> **IMPORTANT**
>
> Block storage volumes, or block persistent volumes, are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.
>
> If you choose to use a block storage volume with the image registry, you must use a filesystem persistent volume claim (PVC).

Procedure

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

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

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

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (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.

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

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

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

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

Example output

```
storage:
  pvc:
    claim: (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.

2.16. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE

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

Prerequisites

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

Procedure

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

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

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False         19m
baremetal                                  4.12.0    True        False         False         37m
cloud-credential                           4.12.0    True        False         False         40m
cluster-autoscaler                         4.12.0    True        False         False         37m
config-operator                            4.12.0    True        False         False         38m
console                                    4.12.0    True        False         False         26m
csi-snapshot-controller                    4.12.0    True        False         False         37m
dns                                        4.12.0    True        False         False         37m
etcd                                       4.12.0    True        False         False         36m
image-registry                             4.12.0    True        False         False         31m
ingress                                    4.12.0    True        False         False         30m
insights                                   4.12.0    True        False         False         31m
kube-apiserver                             4.12.0    True        False         False         26m
kube-controller-manager                    4.12.0    True        False         False         36m
kube-scheduler                             4.12.0    True        False         False         36m
kube-storage-version-migrator              4.12.0    True        False         False         37m
machine-api                                4.12.0    True        False         False         29m
machine-approver                           4.12.0    True        False         False         37m
machine-config                             4.12.0    True        False         False         36m
marketplace                                4.12.0    True        False         False         37m
monitoring                                 4.12.0    True        False         False         29m
network                                    4.12.0    True        False         False         38m
node-tuning                                4.12.0    True        False         False         37m
openshift-apiserver                        4.12.0    True        False         False         32m
openshift-controller-manager               4.12.0    True        False         False         30m
openshift-samples                          4.12.0    True        False         False         32m
operator-lifecycle-manager                 4.12.0    True        False         False         37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False         37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False         32m
service-ca                                 4.12.0    True        False         False         38m
storage                                    4.12.0    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.

IMPORTANT

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

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

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

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

Example output

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

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

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

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

2.17. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.12, 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 Hybrid Cloud Console.

After you confirm that your OpenShift Cluster Manager Hybrid Cloud Console 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

2.18. NEXT STEPS

- Validating an installation.
- Customize your cluster.
- Remote health reporting
- Set up your registry and configure registry storage .

CHAPTER 3. INSTALLING A USER-PROVISIONED BARE METAL CLUSTER WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform 4.12, you can install a cluster on bare metal infrastructure that you provision with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

When you customize OpenShift Container Platform networking, you must set most of the network configuration parameters during installation. You can modify only **kubeProxy** network configuration parameters in a running cluster.

3.1. PREREQUISITES

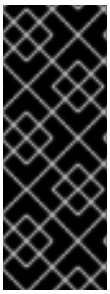
- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- If you use a firewall and plan to use the Telemetry service, you configured the firewall to allow the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

3.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.12, you require access to the internet to install your cluster.

You must have internet access to:

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



IMPORTANT

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

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.

3.3. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

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

3.3.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 3.1. 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.

NOTE

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.

IMPORTANT

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

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

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

3.3.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 3.2. 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

1. 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.

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

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

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

3.3.3. 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.

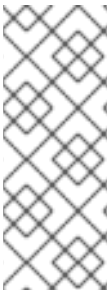
- See Approving the certificate signing requests for your machines for more information about approving cluster certificate signing requests after installation.

3.3.4. 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.



NOTE

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.

3.3.4.1. Setting the cluster node hostnames through DHCP

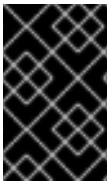
On Red Hat Enterprise Linux CoreOS (RHCOS) machines, the hostname is set through NetworkManager. By default, the machines obtain their hostname through DHCP. If the hostname is not provided by DHCP, set statically through kernel arguments, or another method, it is obtained through a reverse DNS lookup.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

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

3.3.4.2. Network connectivity requirements

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

This section provides details about the ports that are required.



IMPORTANT

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

Table 3.3. 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
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
	123	Network Time Protocol (NTP) on UDP port **123** If an external NTP time server is configured, you must open UDP port **123**.
TCP/UDP	**30000**-**32767**	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 3.4. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 3.5. 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

3.3.5. 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.

NOTE

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

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

Table 3.6. 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.

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

NOTE

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

TIP

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

3.3.5.1. 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.

Example 3.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.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 (6)
control-plane2.ocp4.example.com. IN A 192.168.1.99 (7)
;
compute0.ocp4.example.com. IN A 192.168.1.11 (8)
compute1.ocp4.example.com. IN A 192.168.1.7 (9)
;
;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.
3. 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

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.

NOTE

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

4 Provides name resolution for the bootstrap machine.

5 6 7 Provides name resolution for the control plane machines.

8 9 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.

Example 3.2. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
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. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;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 5 6 Provides reverse DNS resolution for the control plane machines.

7 8 Provides reverse DNS resolution for the compute machines.

NOTE

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

- Validating DNS resolution for user-provisioned infrastructure

3.3.6. Load balancing requirements for user-provisioned infrastructure

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

NOTE

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

The load balancing infrastructure must meet the following requirements:

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

   IMPORTANT

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

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

   Table 3.7. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description

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

NOTE

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

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

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

   TIP

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

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

   Table 3.8. Application Ingress load balancer

Port	Back-end machines (pool members)	Internal	External	Description

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

NOTE

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

3.3.6.1. 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.

NOTE

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

Example 3.3. 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 (4)
  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 (5)
  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 (6)
  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. 4. 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.
5. 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.
6. Port **80** handles the HTTP traffic and points to the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.

NOTE

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

TIP

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

3.4. 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

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

NOTE

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

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

NOTE

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

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

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

IMPORTANT

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

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

4. Setup the required DNS infrastructure for your cluster.

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

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

5. Validate your DNS configuration.

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

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

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

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

NOTE

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

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

3.5. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE

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

IMPORTANT

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

Prerequisites

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

Procedure

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

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

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

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

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

Example output

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

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

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

Example output

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

NOTE

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

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

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

Example output

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

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

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

Example output

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

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

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

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

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

Example output

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

NOTE

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

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

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

Example output

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

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

Additional resources

- User-provisioned DNS requirements
- Load balancing requirements for user-provisioned infrastructure

3.6. 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.

IMPORTANT

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

NOTE

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

Procedure

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

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

(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.

NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS validated or Modules In Process cryptographic libraries on the **x86_64**, **ppc64le**, and **s390x** architectures. do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

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

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

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

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

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

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

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

   Example output

```
Agent pid 31874
```

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

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

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

Additional resources

- Verifying node health

3.7. OBTAINING THE INSTALLATION PROGRAM

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

Prerequisites

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

Procedure

1. 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.

2. Select your infrastructure provider.

3. 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.

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

> **IMPORTANT**
>
> Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

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

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

5. Download your installation pull secret from the 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.

3.8. 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.

> **IMPORTANT**
>
> If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.12. 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

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.12 Linux Client** entry and save the file.
5. Unpack the archive:

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

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

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

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

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.12 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

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

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.12 macOS Client** entry and save the file.

> **NOTE**
>
> For macOS arm64, choose the **OpenShift v4.12 macOS arm64 Client** entry.

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

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

3.9. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE

Installing the cluster requires that you manually create the installation configuration file.

Prerequisites

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

Procedure

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

```
$ mkdir <installation_directory>
```

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

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

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

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

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

3.9.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide a customized **install-config.yaml** installation configuration file that describes the details for your environment.

> **NOTE**
>
> After installation, you cannot modify these parameters in the **install-config.yaml** file.

3.9.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 3.9. Required parameters

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installation program may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the **baseDomain** and **metadata.name** parameter values that uses the **<metadata.name>.<baseDomain>** format.	A fully-qualified domain or subdomain name, such as **example.com**.

Parameter	Description	Values
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.	String of lowercase letters and hyphens (**-**), such as **dev**.
platform	The configuration for the specific platform upon which to perform the installation: **alibabacloud**, **aws**, **baremetal**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**. For additional information about **platform.<platform>** parameters, consult the table for your specific platform that follows.	Object
pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

3.9.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

- If you use the Red Hat OpenShift Networking OVN-Kubernetes network plugin, both IPv4 and IPv6 address families are supported.
- If you use the Red Hat OpenShift Networking OpenShift SDN network plugin, only the IPv4 address family is supported.

If you configure your cluster to use both IP address families, review the following requirements:

- Both IP families must use the same network interface for the default gateway.
- Both IP families must have the default gateway.
- You must specify IPv4 and IPv6 addresses in the same order for all network configuration parameters. For example, in the following configuration IPv4 addresses are listed before IPv6 addresses.

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```

NOTE

Globalnet is not supported with Red Hat OpenShift Data Foundation disaster recovery solutions. For regional disaster recovery scenarios, ensure that you use a nonoverlapping range of private IP addresses for the cluster and service networks in each cluster.

Table 3.10. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object **NOTE** You cannot modify parameters specified by the **networking** object after installation.
networking.networkType	The Red Hat OpenShift Networking network plugin to install.	Either **OpenShiftSDN** or **OVNKubernetes**. **OpenShiftSDN** is a CNI plugin for all-Linux networks. **OVNKubernetes** is a CNI plugin for Linux networks and hybrid networks that contain both Linux and Windows servers. The default value is **OVNKubernetes**.

Parameter	Description	Values

networking.clusterNetwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23
		  - cidr: fd01::/48
		  hostPrefix: 64
		``

networking.clusterNetwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. If you use the OpenShift SDN network plugin, specify an IPv4 network. If you use the OVN-Kubernetes network plugin, you can specify IPv4 and IPv6 networks.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**. The prefix length for an IPv6 block is between **0** and **128**. For example, **10.128.0.0/14** or **fd01::/48**.
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a **/23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 - 23) - 2) pod IP addresses.	A subnet prefix. For an IPv4 network the default value is **23**. For an IPv6 network the default value is **64**. The default value is also the minimum value for IPv6.

networking.serviceNetwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network plugins support only a single IP address block for the service network.
	
	If you use the OVN-Kubernetes network plugin, you can specify an IP address block for both of the IPv4 and IPv6 address families.
		An array with an IP address block in CIDR format. For example:
		
		``
		networking:
		  serviceNetwork:
		  - 172.30.0.0/16
		  - fd02::/112
		``

networking.machineNetwork	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  machineNetwork:
		  - cidr: 10.0.0.0/16
		``

Parameter	Description	Values
networking.machineNetwork.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16** or **fd00::/48**. **NOTE** Set the **networking.machineNetwork** to match the CIDR that the preferred NIC resides in.

3.9.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 3.11. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String
capabilities	Controls the installation of optional core cluster components. You can reduce the footprint of your OpenShift Container Platform cluster by disabling optional components. For more information, see the "Cluster capabilities" page in *Installing*.	String array
capabilities.baselineCapabilitySet	Selects an initial set of optional capabilities to enable. Valid values are **None**, **v4.11**, **v4.12** and **vCurrent**. The default value is **vCurrent**.	String

Parameter	Description	Values
capabilities.additionalEnabledCapabilities	Extends the set of optional capabilities beyond what you specify in **baselineCapabilitySet**. You may specify multiple capabilities in this parameter.	String array
compute	The configuration for the machines that comprise the compute nodes.	Array of **MachinePool** objects.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**alibabacloud**, **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to **2**. The default value is **3**.

Parameter	Description	Values
featureSet	Enables the cluster for a feature set. A feature set is a collection of OpenShift Container Platform features that are not enabled by default. For more information about enabling a feature set during installation, see "Enabling features using feature gates".	String. The name of the feature set to enable, such as **TechPreviewNoUpgrade**.
controlPlane	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.	String
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platform	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**alibabacloud**, **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**

Parameter	Description	Values
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. NOTE Not all CCO modes are supported for all cloud providers. For more information about CCO modes, see the *Cloud Credential Operator* entry in the *Cluster Operators reference* content. NOTE If your AWS account has service control policies (SCP) enabled, you must configure the **credentialsMode** parameter to **Mint**, **Passthrough** or **Manual**.	**Mint**, **Passthrough**, **Manual** or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead. **IMPORTANT** To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64**, **ppc64le**, and **s390x** architectures. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String

Parameter	Description	Values
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. The default value is **External**. Setting this field to **Internal** is not supported on non-cloud platforms. IMPORTANT If the value of the field is set to **Internal**, the cluster will become non-functional. For more information, refer to BZ#1953035.
sshKey	The SSH key to authenticate access to your cluster machines. NOTE For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.	For example, **sshKey: ssh-ed25519 AAAA..**.

3.9.2. 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: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
```

```
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths": ...}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
```

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

2, 5. The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

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

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

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

4. 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.

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

7. 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.

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

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

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

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

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

12. 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.

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

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

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

> **IMPORTANT**
>
> To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64**, **ppc64le**, and **s390x** architectures.

15. The pull secret from the 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.

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

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

Additional resources

- See Load balancing requirements for user-provisioned infrastructure for more information on the API and application ingress load balancing requirements.

3.10. NETWORK CONFIGURATION PHASES

There are two phases prior to OpenShift Container Platform installation where you can customize the network configuration.

Phase 1

You can customize the following network-related fields in the **install-config.yaml** file before you create the manifest files:

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
  For more information on these fields, refer to *Installation configuration parameters*.

> **NOTE**
>
> Set the **networking.machineNetwork** to match the CIDR that the preferred NIC resides in.

> **IMPORTANT**
>
> The CIDR range **172.17.0.0/16** is reserved by libVirt. You cannot use this range or any range that overlaps with this range for any networks in your cluster.

Phase 2

After creating the manifest files by running **openshift-install create manifests**, you can define a customized Cluster Network Operator manifest with only the fields you want to modify. You can use the manifest to specify an advanced network configuration.

You cannot override the values specified in phase 1 in the **install-config.yaml** file during phase 2. However, you can further customize the network plugin during phase 2.

3.11. SPECIFYING ADVANCED NETWORK CONFIGURATION

You can use advanced network configuration for your network plugin to integrate your cluster into your existing network environment. You can specify advanced network configuration only before you install the cluster.

IMPORTANT

Customizing your network configuration by modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

Prerequisites

- You have created the **install-config.yaml** file and completed any modifications to it.

Procedure

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

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

(1) **<installation_directory>** specifies the name of the directory that contains the **install-config.yaml** file for your cluster.

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

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

3. Specify the advanced network configuration for your cluster in the **cluster-network-03-config.yml** file, such as in the following examples:

Specify a different VXLAN port for the OpenShift SDN network provider

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    openshiftSDNConfig:
      vxlanPort: 4800
```

Enable IPsec for the OVN-Kubernetes network provider

```
apiVersion: operator.openshift.io/v1
kind: Network
```

```
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig: {}
```

4. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program consumes the **manifests/** directory when you create the Ignition config files.

3.12. CLUSTER NETWORK OPERATOR CONFIGURATION

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

The CNO configuration inherits the following fields during cluster installation from the **Network** API in the **Network.config.openshift.io** API group and these fields cannot be changed:

clusterNetwork

IP address pools from which pod IP addresses are allocated.

serviceNetwork

IP address pool for services.

defaultNetwork.type

Cluster network plugin, such as OpenShift SDN or OVN-Kubernetes.

You can specify the cluster network plugin configuration for your cluster by setting the fields for the **defaultNetwork** object in the CNO object named **cluster**.

3.12.1. Cluster Network Operator configuration object

The fields for the Cluster Network Operator (CNO) are described in the following table:

Table 3.12. Cluster Network Operator configuration object

Field	Type	Description
metadata.name	**string**	The name of the CNO object. This name is always **cluster**.

Field	Type	Description

spec.clusterNetwork	**array**	A list specifying the blocks of IP addresses from which pod IP addresses are allocated and the subnet prefix length assigned to each individual node in the cluster. For example:
		`spec:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/19`
		`  hostPrefix: 23`
		`  - cidr: 10.128.32.0/19`
		`  hostPrefix: 23`
		You can customize this field only in the **install-config.yaml** file before you create the manifests. The value is read-only in the manifest file.

spec.serviceNetwork	**array**	A block of IP addresses for services. The OpenShift SDN and OVN-Kubernetes network plugins support only a single IP address block for the service network. For example:
		`spec:`
		`  serviceNetwork:`
		`  - 172.30.0.0/14`
		You can customize this field only in the **install-config.yaml** file before you create the manifests. The value is read-only in the manifest file.

spec.defaultNetwork	**object**	Configures the network plugin for the cluster network.
spec.kubeProxyConfig	**object**	The fields for this object specify the kube-proxy configuration. If you are using the OVN-Kubernetes cluster network plugin, the kube-proxy configuration has no effect.

IMPORTANT

For a cluster that needs to deploy objects across multiple networks, ensure that you specify the same value for the **clusterNetwork.hostPrefix** parameter for each network type that is defined in the **install-config.yaml** file. Setting a different value for each **clusterNetwork.hostPrefix** parameter can impact the OVN-Kubernetes network plugin, where the plugin cannot effectively route object traffic among different nodes.

defaultNetwork object configuration

The values for the **defaultNetwork** object are defined in the following table:

Table 3.13. **defaultNetwork** object

Field	Type	Description
type	**string**	Either **OpenShiftSDN** or **OVNKubernetes**. The Red Hat OpenShift Networking network plugin is selected during installation. This value cannot be changed after cluster installation. NOTE OpenShift Container Platform uses the OVN-Kubernetes network plugin by default.
openshiftSDNConfig	**object**	This object is only valid for the OpenShift SDN network plugin.
ovnKubernetesConfig	**object**	This object is only valid for the OVN-Kubernetes network plugin.

Configuration for the OpenShift SDN network plugin

The following table describes the configuration fields for the OpenShift SDN network plugin:

Table 3.14. **openshiftSDNConfig** object

Field	Type	Description
mode	**string**	Configures the network isolation mode for OpenShift SDN. The default value is **NetworkPolicy**. The values **Multitenant** and **Subnet** are available for backwards compatibility with OpenShift Container Platform 3.x but are not recommended. This value cannot be changed after cluster installation.

Field	Type	Description
mtu	**integer**	The maximum transmission unit (MTU) for the VXLAN overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU. If the auto-detected value is not what you expect it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes. If your cluster requires different MTU values for different nodes, you must set this value to **50** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of**1500**, you must set this value to **1450**. You can set the value during cluster installation or as a post-installation task. For more information, see "Changing the MTU for the cluster network" in the OpenShift Container Platform Networking document.
vxlanPort	**integer**	The port to use for all VXLAN packets. The default value is **4789**. This value cannot be changed after cluster installation. If you are running in a virtualized environment with existing nodes that are part of another VXLAN network, then you might be required to change this. For example, when running an OpenShift SDN overlay on top of VMware NSX-T, you must select an alternate port for the VXLAN, because both SDNs use the same default VXLAN port number. On Amazon Web Services (AWS), you can select an alternate port for the VXLAN between port **9000** and port **9999**.

Configuration for the OVN-Kubernetes network plugin

The following table describes the configuration fields for the OVN-Kubernetes network plugin:

Table 3.15. **ovnKubernetesConfig** object

Field	Type	Description

Field	Type	Description
mtu	**integer**	The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU. If the auto-detected value is not what you expect it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes. If your cluster requires different MTU values for different nodes, you must set this value to **100** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of**1500**, you must set this value to **1400**.
genevePort	**integer**	The port to use for all Geneve packets. The default value is **6081**. This value cannot be changed after cluster installation.
ipsecConfig	**object**	Specify an empty object to enable IPsec encryption.
policyAuditConfig	**object**	Specify a configuration object for customizing network policy audit logging. If unset, the defaults audit log settings are used.
gatewayConfig	**object**	Optional: Specify a configuration object for customizing how egress traffic is sent to the node gateway. NOTE While migrating egress traffic, you can expect some disruption to workloads and service traffic until the Cluster Network Operator (CNO) successfully rolls out the changes.

Field	Type	Description
v4InternalSubnet	If your existing network infrastructure overlaps with the **100.64.0.0/16** IPv4 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. You must ensure that the IP address range does not overlap with any other subnet used by your OpenShift Container Platform installation. The IP address range must be larger than the maximum number of nodes that can be added to the cluster. For example, if the **clusterNetwork.cidr** value is **10.128.0.0/14** and the **clusterNetwork.hostPrefix** value is **/23**, then the maximum number of nodes is **2^(23-14)=512**. This field cannot be changed after installation.	The default value is **100.64.0.0/16**.

Field	Type	Description
v6InternalSubnet	If your existing network infrastructure overlaps with the **fd98::/48** IPv6 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. You must ensure that the IP address range does not overlap with any other subnet used by your OpenShift Container Platform installation. The IP address range must be larger than the maximum number of nodes that can be added to the cluster. This field cannot be changed after installation.	The default value is **fd98::/48**.

Table 3.16. **policyAuditConfig** object

Field	Type	Description
rateLimit	integer	The maximum number of messages to generate every second per node. The default value is **20** messages per second.
maxFileSize	integer	The maximum size for the audit log in bytes. The default value is **50000000** or 50 MB.

Field	Type	Description
destination	string	One of the following additional audit log targets: **libc** The libc **syslog()** function of the journald process on the host. **udp:<host>:<port>** A syslog server. Replace **<host>:<port>** with the host and port of the syslog server. **unix:<file>** A Unix Domain Socket file specified by **<file>**. **null** Do not send the audit logs to any additional target.
syslogFacility	string	The syslog facility, such as **kern**, as defined by RFC5424. The default value is **local0**.

Table 3.17. **gatewayConfig** object

Field	Type	Description
routingViaHost	**boolean**	Set this field to **true** to send egress traffic from pods to the host networking stack. NOTE In OpenShift Container Platform 4.12, egress IP is only assigned to the primary interface. Consequentially, setting **routingViaHost** to **true** will not work for egress IP in OpenShift Container Platform 4.12. For highly-specialized installations and applications that rely on manually configured routes in the kernel routing table, you might want to route egress traffic to the host networking stack. By default, egress traffic is processed in OVN to exit the cluster and is not affected by specialized routes in the kernel routing table. The default value is **false**. This field has an interaction with the Open vSwitch hardware offloading feature. If you set this field to **true**, you do not receive the performance benefits of the offloading because egress traffic is processed by the host networking stack.

Example OVN-Kubernetes configuration with IPSec enabled

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
```

```
mtu: 1400
genevePort: 6081
ipsecConfig: {}
```

kubeProxyConfig object configuration

The values for the **kubeProxyConfig** object are defined in the following table:

Table 3.18. **kubeProxyConfig** object

Field	Type	Description
iptablesSyncPeriod	**string**	The refresh period for **iptables** rules. The default value is **30s**. Valid suffixes include **s**, **m**, and **h** and are described in the Go **time** package documentation. **NOTE** Because of performance improvements introduced in OpenShift Container Platform 4.3 and greater, adjusting the **iptablesSyncPeriod** parameter is no longer necessary.

proxyArguments.iptables-min-sync-period	**array**	The minimum duration before refreshing **iptables** rules. This field ensures that the refresh does not happen too frequently. Valid suffixes include **s**, **m**, and **h** and are described in the Go **time** package. The default value is:
		
		`kubeProxyConfig:`
		`  proxyArguments:`
		`  iptables-min-sync-period:`
		`  - 0s`

3.13. CREATING THE IGNITION CONFIG FILES

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

IMPORTANT

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

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

Procedure

- Obtain the Ignition config files:

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

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

IMPORTANT

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

The following files are generated in the directory:

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

3.14. 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.

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

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

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

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

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

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

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

3.14.1. 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

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

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

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

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

   IMPORTANT

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

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

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

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

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

Example output

```
"location": "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live.x86_64.iso",
```

IMPORTANT

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

ISO file names resemble the following example:

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

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

NOTE

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

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

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

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

NOTE

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

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

```
$ sudo coreos-installer install --ignition-
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

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

IMPORTANT

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

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

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

Example command

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

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

IMPORTANT

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

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

NOTE

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

3.14.2. Installing RHCOS by using PXE or iPXE booting

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

Prerequisites

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

Procedure

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

   IMPORTANT

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

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

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

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

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

Example output

```
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-kernel-aarch64"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-initramfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```

IMPORTANT

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

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

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

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

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

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

6. Configure PXE or iPXE installation for the RHCOS images and begin the installation. 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_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (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.

NOTE

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

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

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

1. 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.
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 location of the **initramfs** file that you uploaded to your HTTP server.

NOTE

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

NOTE

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

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

```
menuentry 'Install CoreOS' {
```

```
  linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
  initrd rhcos-<version>-live-initramfs.<architecture>.img (3)
}
```

1. 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.
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 location of the **initramfs** file that you uploaded to your TFTP server.

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

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

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

10. Continue to create the machines for your cluster.

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

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

NOTE

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

3.14.3. 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.

3.14.3.1. Using advanced networking options for PXE and ISO installations

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

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

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

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

To configure an ISO installation, use the following procedure.

Procedure

1. Boot the ISO installer.
2. From the live system shell prompt, configure networking for the live system using available RHEL tools, such as **nmcli** or **nmtui**.

3. Run the **coreos-installer** command to install the system, adding the **--copy-network** option to copy networking configuration. For example:

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

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

4. 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.

3.14.3.2. Disk partitioning

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 you override the default partitioning configuration. During the RHCOS installation, the size of the root file system is increased to use any remaining available space on the target device.

> **IMPORTANT**
>
> The use of a custom partition scheme on your node might result in OpenShift Container Platform not monitoring or alerting on some node partitions. If you override the default partitioning, see Understanding OpenShift File System Monitoring (eviction conditions) for more information about how OpenShift Container Platform monitors your host file systems.

OpenShift Container Platform monitors the following two filesystem identifiers:

- **nodefs**, which is the filesystem that contains **/var/lib/kubelet**
- **imagefs**, which is the filesystem that contains **/var/lib/containers**

For the default partition scheme, **nodefs** and **imagefs** monitor the same root filesystem, /.

To override the default partitioning when installing RHCOS on an OpenShift Container Platform cluster node, you must create separate partitions.

> **IMPORTANT**
>
> 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.

Consider a situation where you want to add a separate storage partition for your containers and container images. For example, by mounting **/var/lib/containers** in a separate partition, the kubelet separately monitors **/var/lib/containers** as the **imagefs** directory and the root file system as the **nodefs** directory.

> **IMPORTANT**
>
> If you have resized your disk size to host a larger file system, consider creating a separate **/var/lib/containers** partition. Consider resizing a disk that has an **xfs** format to reduce CPU time issues caused by a high number of allocation groups.

3.14.3.2.1. 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.

> **IMPORTANT**
>
> 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

1. 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>
```

2. 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: 4.12.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> (1)
    partitions:
    - label: var
      start_mib: <partition_start_offset> (2)
      size_mib: <partition_size> (3)
      number: 5
  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.
2. 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 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.

NOTE

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.

3. 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
```

4. 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.

3.14.3.2.2. 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.

> **NOTE**
>
> 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/sda
```

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

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

This example preserves partitions 5 and higher:

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

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

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
```

3.14.3.3. Identifying Ignition configs

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

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

  > **IMPORTANT**
  >
  > It is not recommended to modify these 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.

3.14.3.4. Default console configuration

Red Hat Enterprise Linux CoreOS (RHCOS) nodes installed from an OpenShift Container Platform 4.12 boot image use a default console that is meant to accomodate 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.

NOTE

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

3.14.3.5. 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

1. Boot the ISO installer.
2. 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/sda
```

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

(2) 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 **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.

3. Reboot into the installed system.

> **NOTE**
>
> 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.

3.14.3.6. 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.

3.14.3.7. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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/sda (2)
```

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

(2) When you specify this option, the ISO image automatically runs an installation. Otherwise, the image remains configured for installation, but does not install automatically unless you specify the **coreos.inst.install_dev** kernel argument.

3. 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.

3.14.3.7.1. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda (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. In this case, **/dev/sda**. 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.

> **NOTE**
>
> 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.

3. 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.

3.14.3.7.2. 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.

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

Procedure

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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
```

> **IMPORTANT**
>
> 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.

3.14.3.7.3. 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.

> **WARNING**
>
> 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto
```

3. 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
slave-type=bond
```

4. 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
slave-type=bond
```

5. 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.

3.14.3.8. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda \ (2)
  -o rhcos-<version>-custom-initramfs.x86_64.img (3)
```

(1) The Ignition config file that is generated from **openshift-installer**.

(2) When you specify this option, the PXE environment automatically runs an install. Otherwise, the image remains 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.

3.14.3.8.1. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda \ (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.

3.14.3.8.2. 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.

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

Procedure

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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
```

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.

> **IMPORTANT**
>
> 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.

3.14.3.8.3. 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.

> **WARNING**
>
> 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto
```

3. 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
slave-type=bond
```

4. 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
slave-type=bond
```

5. 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
```

6. 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.

3.14.3.9. 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.

3.14.3.9.1. 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.

IMPORTANT

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

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

NOTE

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

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

NOTE

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

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.

NOTE

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

- 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 network interfaces to a single interface
Optional: You can configure VLANs on bonded interfaces by using the **vlan=** parameter and to use DHCP, for example:

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

Use the following example to configure the bonded interface with a VLAN and to use a static IP address:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

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**).

NOTE

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

Use the following example to configure a network team:

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

3.14.3.9.2. **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 3.19. coreos-installer subcommands, command-line options, and arguments

coreos-installer install subcommand	
Subcommand	*Description*
$ coreos-installer install <options> <device>	Embed an Ignition config in an ISO image.
coreos-installer install subcommand options	
Option	*Description*
-u, --image-url <url>	Specify the image URL manually.
-f, --image-file <path>	Specify a local image file manually. Used for debugging.
-i, --ignition-file <path>	Embed an Ignition config from a file.
-I, --ignition-url <URL>	Embed an Ignition config from a URL.
--ignition-hash <digest>	Digest **type-value** of the Ignition config.
-p, --platform <name>	Override the Ignition platform ID for the installed system.
--console <spec>	Set the kernel and bootloader console for the installed system. For more information about the format of **<spec>**, see the Linux kernel serial console documentation.
--append-karg <arg>...	Append a default kernel argument to the installed system.
--delete-karg <arg>...	Delete a default kernel argument from the installed system.

-n, **--copy-network**	Copy the network configuration from the install environment. IMPORTANT The **--copy-network** option only copies networking configuration found under **/etc/NetworkManager/system-connections**. In particular, it does not copy the system hostname.
--network-dir <path>	For use with **-n**. Default is **/etc/NetworkManager/system-connections/**.
--save-partlabel <lx>..	Save partitions with this label glob.
--save-partindex <id>…	Save partitions with this number or range.
--insecure	Skip RHCOS image signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Valid values are **x86_64** and **aarch64**.
--preserve-on-error	Do not clear partition table on error.
-h, **--help**	Print help information.
coreos-installer install subcommand argument	
Argument	*Description*
<device>	The destination device.
coreos-installer ISO subcommands	
Subcommand	*Description*
$ coreos-installer iso customize <options> <ISO_image>	Customize a RHCOS live ISO image.

coreos-installer iso reset <options> <ISO_image>	Restore a RHCOS live ISO image to default settings.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.
coreos-installer ISO customize subcommand options	
Option	*Description*
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and 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	
Subcommand	*Description*
Note that not all of these options are accepted by all subcommands.	
coreos-installer pxe customize <options> <path>	Customize a RHCOS live PXE boot config.
coreos-installer pxe ignition wrap <options>	Wrap an Ignition config in an image.
coreos-installer pxe ignition unwrap <options> <image_name>	Show the wrapped Ignition config in an image.
coreos-installer PXE customize subcommand options	
Option	*Description*
Note that not all of these options are accepted by all subcommands.	
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and 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.
post-install <path>	Run the specified script after installation.
--installer-config <path>	Apply the specified installer configuration file.

--live-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
-o, --output <path>	Write the initramfs to a new output file. **NOTE** This option is required for PXE environments.
-h, --help	Print help information.

3.14.3.9.3. 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 3.20. **coreos.inst** boot options

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to. It is recommended to use the full path, such as **/dev/sda**, although **sda** is allowed.
coreos.inst.ignition_url	Optional: The URL of the Ignition config to embed into the installed system. If no URL is specified, no Ignition config is embedded. Only HTTP and HTTPS protocols are supported.
coreos.inst.save_partlabel	Optional: Comma-separated labels of partitions to preserve during the install. Glob-style wildcards are permitted. The specified partitions do not need to exist.
coreos.inst.save_partindex	Optional: Comma-separated indexes of partitions to preserve during the install. Ranges **m-n** are permitted, and either **m** or **n** can be omitted. The specified partitions do not need to exist.

Argument	Description
coreos.inst.insecure	Optional: Permits the OS image that is specified by **coreos.inst.image_url** to be unsigned.
coreos.inst.image_url	Optional: Download and install the specified RHCOS image. • This argument should not be used in production environments and is intended for debugging purposes only. • While this argument can be used to install a version of RHCOS that does not match the live media, it is recommended that you instead use the media that matches the version you want to install. • If you are using **coreos.inst.image_url**, you must also use **coreos.inst.insecure**. This is because the bare-metal media are not GPG-signed for OpenShift Container Platform. • Only HTTP and HTTPS protocols are supported.
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. After the install finishes, you will receive a prompt that allows you to inspect what is happening during installation. This argument should not be used in production environments and is intended for debugging purposes only.
coreos.inst.platform_id	Optional: The Ignition platform ID of the platform the RHCOS image is being installed on. Default is **metal**. This option determines whether or not to request an Ignition config from the cloud provider, such as VMware. For example: **coreos.inst.platform_id=vmware**.
ignition.config.url	Optional: The URL of the Ignition config for the live boot. For example, this can be used to customize how **coreos-installer** is invoked, or to run code before or after the installation. This is different from **coreos.inst.ignition_url**, which is the Ignition config for the installed system.

3.14.4. 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.

IMPORTANT

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.

NOTE

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

Prerequisites

- You have created the Ignition config files for your cluster.
- You have reviewed *Installing RHCOS and starting the OpenShift Container Platform bootstrap process*.

Procedure

1. To enable multipath and start the **multipathd** daemon, run the following command on the installation host:

```
$ 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.

2. 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)
--ignition-url=http://host/worker.ign \
--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)
--ignition-url=http://host/worker.ign \
--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".

3. Reboot into the installed system.

4. 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.

3.14.4.1. Enabling multipathing on secondary disks

RHCOS also supports multipathing on a secondary disk. Instead of kernel arguments, you use Ignition to enable multipathing for the secondary disk at installation time.

Prerequisites

- You have read the section *Disk partitioning*.
- You have read *Enabling multipathing with kernel arguments on RHCOS* .
- You have installed the Butane utility.

Procedure

1. Create a Butane config with information similar to the following:

Example multipath-config.bu

```
variant: openshift
version: 4.12.0
systemd:
  units:
    - name: mpath-configure.service
      enabled: true
      contents: |
        [Unit]
        Description=Configure Multipath on Secondary Disk
        ConditionFirstBoot=true
        ConditionPathExists=!/etc/multipath.conf
        Before=multipathd.service (1)
        DefaultDependencies=no

        [Service]
        Type=oneshot
        ExecStart=/usr/sbin/mpathconf --enable (2)

        [Install]
        WantedBy=multi-user.target
    - name: mpath-var-lib-container.service
      enabled: true
      contents: |
        [Unit]
        Description=Set Up Multipath On /var/lib/containers
        ConditionFirstBoot=true (3)
        Requires=dev-mapper-mpatha.device
        After=dev-mapper-mpatha.device
        After=ostree-remount.service
        Before=kubelet.service
        DefaultDependencies=no

        [Service] (4)
        Type=oneshot
        ExecStart=/usr/sbin/mkfs.xfs -L containers -m reflink=1 /dev/mapper/mpatha
        ExecStart=/usr/bin/mkdir -p /var/lib/containers

        [Install]
        WantedBy=multi-user.target
    - name: var-lib-containers.mount
      enabled: true
      contents: |
        [Unit]
        Description=Mount /var/lib/containers
        After=mpath-var-lib-containers.service
        Before=kubelet.service (5)

        [Mount] (6)
        What=/dev/disk/by-label/dm-mpath-containers
        Where=/var/lib/containers
        Type=xfs
```

```
[Install]
WantedBy=multi-user.target
```

1. The configuration must be set before launching the multipath daemon.
2. Starts the **mpathconf** utility.
3. This field must be set to the value **true**.
4. Creates the filesystem and directory **/var/lib/containers**.
5. The device must be mounted before starting any nodes.
6. Mounts the device to the **/var/lib/containers** mount point. This location cannot be a symlink.

2. Create the Ignition configuration by running the following command:

```
$ butane --pretty --strict multipath-config.bu > multipath-config.ign
```

3. Continue with the rest of the first boot RHCOS installation process.

IMPORTANT

Do not add the **rd.multipath** or **root** kernel arguments on the command-line during installation unless the primary disk is also multipathed.

3.15. 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

1. Monitor the bootstrap process:

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

1. For **<installation_directory>**, specify the path to the directory that you stored the installation files in.
2. To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.25.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

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

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

IMPORTANT

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

Additional resources

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

3.16. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

3.17. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

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

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.25.0
master-1  Ready     master  63m  v1.25.0
master-2  Ready     master  64m  v1.25.0
```

The output lists all of the machines that you created.

NOTE

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

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

```
$ oc get csr
```

Example output

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

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

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

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

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

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

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

```
$ oc adm certificate approve <csr_name> (1)
```

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

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

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

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

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

```
$ oc get csr
```

Example output

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

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

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

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

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

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

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

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

   ```
   $ oc get nodes
   ```

   Example output

   ```
   NAME      STATUS   ROLES    AGE  VERSION
   master-0  Ready    master   73m  v1.25.0
   master-1  Ready    master   73m  v1.25.0
   master-2  Ready    master   74m  v1.25.0
   worker-0  Ready    worker   11m  v1.25.0
   worker-1  Ready    worker   11m  v1.25.0
   ```

   NOTE

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

Additional information

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

3.18. INITIAL OPERATOR CONFIGURATION

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

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False      19m
baremetal                                  4.12.0    True        False         False      37m
cloud-credential                           4.12.0    True        False         False      40m
cluster-autoscaler                         4.12.0    True        False         False      37m
config-operator                            4.12.0    True        False         False      38m
console                                    4.12.0    True        False         False      26m
csi-snapshot-controller                    4.12.0    True        False         False      37m
dns                                        4.12.0    True        False         False      37m
etcd                                       4.12.0    True        False         False      36m
image-registry                             4.12.0    True        False         False      31m
ingress                                    4.12.0    True        False         False      30m
insights                                   4.12.0    True        False         False      31m
kube-apiserver                             4.12.0    True        False         False      26m
kube-controller-manager                    4.12.0    True        False         False      36m
kube-scheduler                             4.12.0    True        False         False      36m
kube-storage-version-migrator              4.12.0    True        False         False      37m
machine-api                                4.12.0    True        False         False      29m
machine-approver                           4.12.0    True        False         False      37m
machine-config                             4.12.0    True        False         False      36m
marketplace                                4.12.0    True        False         False      37m
monitoring                                 4.12.0    True        False         False      29m
network                                    4.12.0    True        False         False      38m
node-tuning                                4.12.0    True        False         False      37m
openshift-apiserver                        4.12.0    True        False         False      32m
openshift-controller-manager               4.12.0    True        False         False      30m
openshift-samples                          4.12.0    True        False         False      32m
operator-lifecycle-manager                 4.12.0    True        False         False      37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False      37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False      32m
service-ca                                 4.12.0    True        False         False      38m
storage                                    4.12.0    True        False         False      37m
```

2. 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.

3.18.1. Image registry removed during installation

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

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

3.18.2. Image registry storage configuration

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

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

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

3.18.3. Configuring block registry storage for bare metal

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

IMPORTANT

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

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

Procedure

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

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

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

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (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.

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

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

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

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

Example output

```
storage:
  pvc:
    claim: (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.

3.19. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE

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

Prerequisites

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

Procedure

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

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

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False         19m
baremetal                                  4.12.0    True        False         False         37m
cloud-credential                           4.12.0    True        False         False         40m
cluster-autoscaler                         4.12.0    True        False         False         37m
config-operator                            4.12.0    True        False         False         38m
console                                    4.12.0    True        False         False         26m
csi-snapshot-controller                    4.12.0    True        False         False         37m
dns                                        4.12.0    True        False         False         37m
etcd                                       4.12.0    True        False         False         36m
image-registry                             4.12.0    True        False         False         31m
ingress                                    4.12.0    True        False         False         30m
insights                                   4.12.0    True        False         False         31m
kube-apiserver                             4.12.0    True        False         False         26m
kube-controller-manager                    4.12.0    True        False         False         36m
kube-scheduler                             4.12.0    True        False         False         36m
kube-storage-version-migrator              4.12.0    True        False         False         37m
machine-api                                4.12.0    True        False         False         29m
machine-approver                           4.12.0    True        False         False         37m
machine-config                             4.12.0    True        False         False         36m
marketplace                                4.12.0    True        False         False         37m
monitoring                                 4.12.0    True        False         False         29m
network                                    4.12.0    True        False         False         38m
node-tuning                                4.12.0    True        False         False         37m
openshift-apiserver                        4.12.0    True        False         False         32m
openshift-controller-manager               4.12.0    True        False         False         30m
openshift-samples                          4.12.0    True        False         False         32m
operator-lifecycle-manager                 4.12.0    True        False         False         37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False         37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False         32m
service-ca                                 4.12.0    True        False         False         38m
storage                                    4.12.0    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.

IMPORTANT

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

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

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

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

   #### Example output

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

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

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

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

3.20. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.12, 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 Hybrid Cloud Console.

After you confirm that your OpenShift Cluster Manager Hybrid Cloud Console 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

3.21. NEXT STEPS

- Validating an installation.
- Customize your cluster.
- Remote health reporting
- Set up your registry and configure registry storage .

CHAPTER 4. INSTALLING A USER-PROVISIONED BARE METAL CLUSTER ON A RESTRICTED NETWORK

In OpenShift Container Platform 4.12, you can install a cluster on bare metal infrastructure that you provision in a restricted network.

IMPORTANT

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

4.1. PREREQUISITES

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You created a registry on your mirror host and obtained the **imageContentSources** data for your version of OpenShift Container Platform.

  IMPORTANT

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

- You provisioned persistent storage for your cluster. To deploy a private image registry, your storage must provide ReadWriteMany access modes.
- If you use a firewall and plan to use the Telemetry service, you configured the firewall to allow the sites that your cluster requires access to.

  NOTE

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

4.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS

In OpenShift Container Platform 4.12, you can perform an installation that does not require an active connection to the internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's Route 53 DNS and IAM services, require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware, Nutanix, or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the

OpenShift image registry and contains the installation media. You can create this registry on a mirror host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.

IMPORTANT

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

4.2.1. Additional limits

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

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

4.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.12, you require access to the internet to obtain the images that are necessary to install your cluster.

You must have internet access to:

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

IMPORTANT

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

4.4. 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.

4.4.1. Required machines for cluster installation

The smallest OpenShift Container Platform clusters require the following hosts:

Table 4.1. 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.

NOTE

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.

IMPORTANT

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

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

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

4.4.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 4.2. Minimum resource requirements

Machine	Operating System	CPU [1]	RAM	Storage	Input/Output Per Second (IOPS)[2]
Bootstrap	RHCOS	4	16 GB	100 GB	300

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

1. One 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.

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

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

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

4.4.3. 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.

4.4.4. 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.

NOTE

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.

4.4.4.1. Setting the cluster node hostnames through DHCP

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

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

4.4.4.2. Network connectivity requirements

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

This section provides details about the ports that are required.

Table 4.3. Ports used for all-machine to all-machine communications

Protocol	Port	Description
ICMP	N/A	Network reachability tests

Protocol	Port	Description
TCP	**1936**	Metrics
	9000-9999	Host level services, including the node exporter on ports **9100-9101** and the Cluster Version Operator on port**9099**.
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	**4789**	VXLAN
	6081	Geneve
	9000-9999	Host level services, including the node exporter on ports **9100-9101**.
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
	123	Network Time Protocol (NTP) on UDP port **123** If an external NTP time server is configured, you must open UDP port **123**.
TCP/UDP	**30000-32767**	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 4.4. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 4.5. 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

4.4.5. 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.

NOTE

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

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

Table 4.6. 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.

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

NOTE

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

TIP

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

4.4.5.1. 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.

Example 4.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.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 (6)
control-plane2.ocp4.example.com. IN A 192.168.1.99 (7)
;
compute0.ocp4.example.com. IN A 192.168.1.11 (8)
compute1.ocp4.example.com. IN A 192.168.1.7 (9)
;
;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.
3. 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

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.

NOTE

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

4 Provides name resolution for the bootstrap machine.

5 6 7 Provides name resolution for the control plane machines.

8 9 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.

Example 4.2. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
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. 5
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com. 6
;
11.1.168.192.in-addr.arpa. IN PTR compute0.ocp4.example.com. 7
7.1.168.192.in-addr.arpa. IN PTR compute1.ocp4.example.com. 8
;
;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 5 6 Provides reverse DNS resolution for the control plane machines.

7 8 Provides reverse DNS resolution for the compute machines.

NOTE

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

Additional resources

- Validating DNS resolution for user-provisioned infrastructure

4.4.6. Load balancing requirements for user-provisioned infrastructure

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

NOTE

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

The load balancing infrastructure must meet the following requirements:

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

   IMPORTANT

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

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

   Table 4.7. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description

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

NOTE

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

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

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

TIP

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

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

Table 4.8. 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

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

NOTE

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

4.4.6.1. 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.

NOTE

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

Example 4.3. 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 (4)
  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 (5)
  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 (6)
  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. 4 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.
5. 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.
6. Port **80** handles the HTTP traffic and points to the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default.

NOTE

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

■

TIP

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

4.5. 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

1. If you are using DHCP to provide the IP networking configuration to your cluster nodes, configure your DHCP service.

   a. Add persistent IP addresses for the nodes to your DHCP server configuration. In your configuration, match the MAC address of the relevant network interface to the intended IP address for each node.

   b. When you use DHCP to configure IP addressing for the cluster machines, the machines also obtain the DNS server information through DHCP. Define the persistent DNS server address that is used by the cluster nodes through your DHCP server configuration.

   NOTE

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

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

NOTE

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

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

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

IMPORTANT

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

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

4. Setup the required DNS infrastructure for your cluster.

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

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

5. Validate your DNS configuration.

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

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

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

NOTE

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

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

4.6. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE

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

IMPORTANT

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

Prerequisites

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

Procedure

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

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

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

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

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

Example output

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

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

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

Example output

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

NOTE

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

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

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

Example output

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

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

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

Example output

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

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

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

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

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

Example output

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

NOTE

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

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

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

Example output

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

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

Additional resources

- User-provisioned DNS requirements
- Load balancing requirements for user-provisioned infrastructure

4.7. 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.

IMPORTANT

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

NOTE

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

Procedure

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

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

(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.

NOTE

If you plan to install an OpenShift Container Platform cluster that uses FIPS validated or Modules In Process cryptographic libraries on the **x86_64**, **ppc64le**, and **s390x** architectures. do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

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

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

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

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

NOTE

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

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

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

Example output

```
Agent pid 31874
```

NOTE

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

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

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

4.8. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE

Installing the cluster requires that you manually create the installation configuration file.

Prerequisites

- You have an SSH public key on your local machine for use with the installation program. You can use the key for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.
- Obtain the **imageContentSources** section from the output of the command to mirror the repository.
- Obtain the contents of the certificate for your mirror registry.

Procedure

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

```
$ mkdir <installation_directory>
```

–

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

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

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

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

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

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

4.8.1. Installation configuration parameters

Before you deploy an OpenShift Container Platform cluster, you provide a customized **install-config.yaml** installation configuration file that describes the details for your environment.

> **NOTE**
>
> After installation, you cannot modify these parameters in the **install-config.yaml** file.

4.8.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 4.9. Required parameters

Parameter	Description	Values

Parameter	Description	Values
apiVersion	The API version for the **install-config.yaml** content. The current version is **v1**. The installation program may also support older API versions.	String
baseDomain	The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the **baseDomain** and **metadata.name** parameter values that uses the **<metadata.name>.<baseDomain>** format.	A fully-qualified domain or subdomain name, such as **example.com**.
metadata	Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.	Object
metadata.name	The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.	String of lowercase letters and hyphens (**-**), such as **dev**.
platform	The configuration for the specific platform upon which to perform the installation: **alibabacloud**, **aws**, **baremetal**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**. For additional information about **platform.<platform>** parameters, consult the table for your specific platform that follows.	Object

Parameter	Description	Values
pullSecret	Get a pull secret from the Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }`

4.8.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or provide different IP address blocks than the defaults.

- If you use the Red Hat OpenShift Networking OVN-Kubernetes network plugin, both IPv4 and IPv6 address families are supported.
- If you use the Red Hat OpenShift Networking OpenShift SDN network plugin, only the IPv4 address family is supported.

If you configure your cluster to use both IP address families, review the following requirements:

- Both IP families must use the same network interface for the default gateway.
- Both IP families must have the default gateway.
- You must specify IPv4 and IPv6 addresses in the same order for all network configuration parameters. For example, in the following configuration IPv4 addresses are listed before IPv6 addresses.

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  - cidr: fd00:10:128::/56
    hostPrefix: 64
  serviceNetwork:
  - 172.30.0.0/16
  - fd00:172:16::/112
```

NOTE

Globalnet is not supported with Red Hat OpenShift Data Foundation disaster recovery solutions. For regional disaster recovery scenarios, ensure that you use a nonoverlapping range of private IP addresses for the cluster and service networks in each cluster.

Table 4.10. Network parameters

Parameter	Description	Values
networking	The configuration for the cluster network.	Object **NOTE** You cannot modify parameters specified by the **networking** object after installation.
networking.networkType	The Red Hat OpenShift Networking network plugin to install.	Either **OpenShiftSDN** or **OVNKubernetes**. **OpenShiftSDN** is a CNI plugin for all-Linux networks. **OVNKubernetes** is a CNI plugin for Linux networks and hybrid networks that contain both Linux and Windows servers. The default value is **OVNKubernetes**.

networking.clusterNetwork	The IP address blocks for pods.
	
	The default value is **10.128.0.0/14** with a host prefix of **/23**.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		  - cidr: fd01::/48
		    hostPrefix: 64
		``

networking.clusterNetwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. If you use the OpenShift SDN network plugin, specify an IPv4 network. If you use the OVN-Kubernetes network plugin, you can specify IPv4 and IPv6 networks.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**. The prefix length for an IPv6 block is between **0** and **128**. For example, **10.128.0.0/14** or **fd01::/48**.
networking.clusterNetwork.hostPrefix	The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a **/23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 - 23) - 2) pod IP addresses.	A subnet prefix. For an IPv4 network the default value is **23**. For an IPv6 network the default value is **64**. The default value is also the minimum value for IPv6.

Parameter	Description	Values

networking.serviceNetwork	The IP address block for services. The default value is **172.30.0.0/16**.
	
	The OpenShift SDN and OVN-Kubernetes network plugins support only a single IP address block for the service network.
	
	If you use the OVN-Kubernetes network plugin, you can specify an IP address block for both of the IPv4 and IPv6 address families.
		An array with an IP address block in CIDR format. For example:
		
		``
		networking:
		  serviceNetwork:
		  - 172.30.0.0/16
		  - fd02::/112
		``

networking.machineNetwork	The IP address blocks for machines.
	
	If you specify multiple IP address blocks, the blocks must not overlap.
		An array of objects. For example:
		
		``
		networking:
		  machineNetwork:
		  - cidr: 10.0.0.0/16
		``

networking.machineNetwork.cidr	Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt. For libvirt, the default value is **192.168.126.0/24**.	An IP network block in CIDR notation. For example, **10.0.0.0/16** or **fd00::/48**. **NOTE** Set the **networking.machineNetwork** to match the CIDR that the preferred NIC resides in.

4.8.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 4.11. Optional parameters

Parameter	Description	Values
additionalTrustBundle	A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle may also be used when a proxy has been configured.	String

Parameter	Description	Values
capabilities	Controls the installation of optional core cluster components. You can reduce the footprint of your OpenShift Container Platform cluster by disabling optional components. For more information, see the "Cluster capabilities" page in *Installing*.	String array
capabilities.baselineCapabilitySet	Selects an initial set of optional capabilities to enable. Valid values are **None**, **v4.11**, **v4.12** and **vCurrent**. The default value is **vCurrent**.	String
capabilities.additionalEnabledCapabilities	Extends the set of optional capabilities beyond what you specify in **baselineCapabilitySet**. You may specify multiple capabilities in this parameter.	String array
compute	The configuration for the machines that comprise the compute nodes.	Array of **MachinePool** objects.
compute.architecture	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.	String
compute.hyperthreading	Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. IMPORTANT If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**

Parameter	Description	Values
compute.name	Required if you use **compute**. The name of the machine pool.	**worker**
compute.platform	Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.	**alibabacloud**, **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**
compute.replicas	The number of compute machines, which are also known as worker machines, to provision.	A positive integer greater than or equal to **2**. The default value is **3**.
featureSet	Enables the cluster for a feature set. A feature set is a collection of OpenShift Container Platform features that are not enabled by default. For more information about enabling a feature set during installation, see "Enabling features using feature gates".	String. The name of the feature set to enable, such as **TechPreviewNoUpgrade**.
controlPlane	The configuration for the machines that comprise the control plane.	Array of **MachinePool** objects.
controlPlane.architecture	Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.	String

Parameter	Description	Values
controlPlane.hyperthreading	Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.	**Enabled** or **Disabled**
controlPlane.name	Required if you use **controlPlane**. The name of the machine pool.	**master**
controlPlane.platform	Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.	**alibabacloud**, **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **ovirt**, **vsphere**, or **{}**
controlPlane.replicas	The number of control plane machines to provision.	The only supported value is **3**, which is the default value.

Parameter	Description	Values
credentialsMode	The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported. NOTE Not all CCO modes are supported for all cloud providers. For more information about CCO modes, see the *Cloud Credential Operator* entry in the *Cluster Operators reference* content. NOTE If your AWS account has service control policies (SCP) enabled, you must configure the **credentialsMode** parameter to **Mint**, **Passthrough** or **Manual**.	**Mint**, **Passthrough**, **Manual** or an empty string (**""**).

Parameter	Description	Values
fips	Enable or disable FIPS mode. The default is **false** (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead. **IMPORTANT** To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64**, **ppc64le**, and **s390x** architectures. **NOTE** If you are using Azure File storage, you cannot enable FIPS mode.	**false** or **true**

Parameter	Description	Values
imageContentSources	Sources and repositories for the release-image content.	Array of objects. Includes a **source** and, optionally, **mirrors**, as described in the following rows of this table.
imageContentSources.source	Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.	String
imageContentSources.mirrors	Specify one or more repositories that may also contain the same images.	Array of strings
publish	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	**Internal** or **External**. The default value is **External**. Setting this field to **Internal** is not supported on non-cloud platforms. IMPORTANT If the value of the field is set to **Internal**, the cluster will become non-functional. For more information, refer to BZ#1953035.
sshKey	The SSH key to authenticate access to your cluster machines. NOTE For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.	For example, **sshKey: ssh-ed25519 AAAA..**.

4.8.2. 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: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OVNKubernetes (11)
  serviceNetwork: (12)
  - 172.30.0.0/16
platform:
  none: {} (13)
fips: false (14)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (15)
sshKey: 'ssh-ed25519 AAAA...' (16)
additionalTrustBundle: | (17)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

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

2, 5. The **controlPlane** section is a single mapping, but the **compute** section is a sequence of mappings. To meet the requirements of the different data structures, the first line of the **compute** section must begin with a hyphen, **-**, and the first line of the **controlPlane** section must not. Only one control plane pool is used.

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

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

IMPORTANT

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

4. 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.

NOTE

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

7. 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.

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

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

NOTE

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

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

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

12. 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.

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

IMPORTANT

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

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

IMPORTANT

To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode. The use of FIPS validated or Modules In Process cryptographic libraries is only supported on OpenShift Container Platform deployments on the **x86_64**, **ppc64le**, and **s390x** architectures.

15. For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

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

NOTE

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

17. Provide the contents of the certificate file that you used for your mirror registry.

18. Provide the **imageContentSources** section from the output of the command to mirror the repository.

Additional resources

- See Load balancing requirements for user-provisioned infrastructure for more information on the API and application ingress load balancing requirements.

4.8.3. Configuring the cluster-wide proxy during installation

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

NOTE

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

Prerequisites

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

NOTE

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

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

Procedure

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

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> (1)
  httpsProxy: 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.

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

(4)

If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates

5. 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**.

NOTE

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

NOTE

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

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

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

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

NOTE

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

4.8.4. 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
```

NOTE

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

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

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

4.9. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES

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

IMPORTANT

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

Prerequisites

- You obtained the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.
- You created the **install-config.yaml** installation configuration file.

Procedure

1. Change to the directory that contains the 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.

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

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

2. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:
   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.
   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.
   c. Save and exit the file.
3. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

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

4.10. CONFIGURING CHRONY TIME SERVICE

You must set the time server and related settings used by the chrony time service (**chronyd**) by modifying the contents of the **chrony.conf** file and passing those contents to your nodes as a machine config.

Procedure

1. Create a Butane config including the contents of the **chrony.conf** file. For example, to configure chrony on worker nodes, create a **99-worker-chrony.bu** file.

> **NOTE**
>
> The Butane version you specify in the config file should match the OpenShift Container Platform version and always ends in **0**. For example, **4.12.0**. See "Creating machine configs with Butane" for information about Butane.

```
variant: openshift
version: 4.12.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644 (3)
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst (4)
        driftfile /var/lib/chrony/drift
```

```
makestep 1.0 3
rtcsync
logdir /var/log/chrony
```

1 2 On control plane nodes, substitute **master** for **worker** in both of these locations.

3 Specify an octal value mode for the **mode** field in the machine config file. After creating the file and applying the changes, the **mode** is converted to a decimal value. You can check the YAML file with the command **oc get mc <mc-name> -o yaml**.

4 Specify any valid, reachable time source, such as the one provided by your DHCP server.

2. Use Butane to generate a **MachineConfig** object file, **99-worker-chrony.yaml**, containing the configuration to be delivered to the nodes:

```
$ butane 99-worker-chrony.bu -o 99-worker-chrony.yaml
```

3. Apply the configurations in one of two ways:

   - If the cluster is not running yet, after you generate manifest files, add the **MachineConfig** object file to the **<installation_directory>/openshift** directory, and then continue to create the cluster.

   - If the cluster is already running, apply the file:

     ```
     $ oc apply -f ./99-worker-chrony.yaml
     ```

4.11. 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.

NOTE

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

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

- Kernel arguments: You can use kernel arguments to provide installation-specific information. For example, you can specify the locations of the RHCOS installation files that you uploaded to your HTTP server and the location of the Ignition config file for the type of node you are installing. For a PXE installation, you can use the **APPEND** parameter to pass the arguments to

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

- Ignition configs: OpenShift Container Platform Ignition config files (***.ign**) are specific to the type of node you are installing. You pass the location of a bootstrap, control plane, or compute node Ignition config file during the RHCOS installation so that it takes effect on first boot. In special cases, you can create a separate, limited Ignition config to pass to the live system. That Ignition config could do a certain set of tasks, such as reporting success to a provisioning system after completing installation. This special Ignition config is consumed by the **coreos-installer** to be applied on first boot of the installed system. Do not provide the standard control plane and compute node Ignition configs to the live ISO directly.
- **coreos-installer**: You can boot the live ISO installer to a shell prompt, which allows you to prepare the permanent system in a variety of ways before first boot. In particular, you can run the **coreos-installer** command to identify various artifacts to include, work with disk partitions, and set up networking. In some cases, you can configure features on the live system and copy them to the installed system.

NOTE

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

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

4.11.1. 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

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

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

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

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

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

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

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

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

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

Example output

```
"location": "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-
<release>-live.aarch64.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-
<release>-live.ppc64le.iso",
"location": "<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-
live.s390x.iso",
"location": "<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-
live.x86_64.iso",
```

IMPORTANT

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

ISO file names resemble the following example:

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

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

NOTE

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

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

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

(1) (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.

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

NOTE

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

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

```
$ sudo coreos-installer install --ignition-
```

```
url=http://192.168.1.2:80/installation_directory/bootstrap.ign /dev/sda --ignition-hash=sha512-
a5a2d43879223273c9b60af66b44202a1d1248fc01cf156c46d4a79f552b6bad47bc8cc78ddf011
6e80c59d2ea9e32ba53bc807afbca581aa059311def2c3e3b
```

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

IMPORTANT

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

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

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

Example command

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

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

IMPORTANT

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

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

NOTE

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

4.11.2. Installing RHCOS by using PXE or iPXE booting

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

Prerequisites

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

Procedure

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

   IMPORTANT

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

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

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

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

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

   Example output

   ```
   "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-
   kernel-aarch64"
   "<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-
   initramfs.aarch64.img"
   ```

```
"<url>/art/storage/releases/rhcos-4.12-aarch64/<release>/aarch64/rhcos-<release>-live-rootfs.aarch64.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/49.84.202110081256-0/ppc64le/rhcos-<release>-live-kernel-ppc64le"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-initramfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-ppc64le/<release>/ppc64le/rhcos-<release>-live-rootfs.ppc64le.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-kernel-s390x"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-initramfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12-s390x/<release>/s390x/rhcos-<release>-live-rootfs.s390x.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-kernel-x86_64"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-initramfs.x86_64.img"
"<url>/art/storage/releases/rhcos-4.12/<release>/x86_64/rhcos-<release>-live-rootfs.x86_64.img"
```

IMPORTANT

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

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

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

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

IMPORTANT

If you plan to add more compute machines to your cluster after you finish installation, do not delete these files.

5. Configure the network boot infrastructure so that the machines boot from their local disks after RHCOS is installed on them.

6. Configure PXE or iPXE installation for the RHCOS images and begin the installation. 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_url=http://<HTTP_server>/rhcos-<version>-live-
rootfs.<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2) (3)
```

(1) (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.

> **NOTE**
>
> This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **APPEND** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? and "Enabling the serial console for PXE and ISO installation" in the "Advanced RHCOS installation configuration" section.

- For iPXE (**x86_64** + **aarch64**):

```
kernel http://<HTTP_server>/rhcos-<version>-live-kernel-<architecture> initrd=main
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
initrd --name main http://<HTTP_server>/rhcos-<version>-live-initramfs.
<architecture>.img (3)
boot
```

(1) 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.

(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 location of the **initramfs** file that you uploaded to your HTTP server.

> **NOTE**
>
> This configuration does not enable serial console access on machines with a graphical console. To configure a different console, add one or more **console=** arguments to the **kernel** line. For example, add **console=tty0 console=ttyS0** to set the first PC serial port as the primary console and the graphical console as a secondary console. For more information, see How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? and "Enabling the serial console for PXE and ISO installation" in the "Advanced RHCOS installation configuration" section.

> **NOTE**
>
> To network boot the CoreOS **kernel** on **aarch64** architecture, you need to use a version of iPXE build with the **IMAGE_GZIP** option enabled. See **IMAGE_GZIP** option in iPXE .

- For PXE (with UEFI and Grub as second stage) on **aarch64**:

```
menuentry 'Install CoreOS' {
  linux rhcos-<version>-live-kernel-<architecture>
coreos.live.rootfs_url=http://<HTTP_server>/rhcos-<version>-live-rootfs.
<architecture>.img coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1) (2)
  initrd rhcos-<version>-live-initramfs.<architecture>.img (3)
}
```

1. 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.
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 location of the **initramfs** file that you uploaded to your TFTP server.

7. Monitor the progress of the RHCOS installation on the console of the machine.

> **IMPORTANT**
>
> Be sure that the installation is successful on each node before commencing with the OpenShift Container Platform installation. Observing the installation process can also help to determine the cause of RHCOS installation issues that might arise.

8. After RHCOS installs, the system reboots. During reboot, the system applies the Ignition config file that you specified.

9. 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
   ```

10. Continue to create the machines for your cluster.

   IMPORTANT

   You must create the bootstrap and control plane machines at this time. If the control plane machines are not made schedulable, also create at least two compute machines before you install the cluster.

   If the required network, DNS, and load balancer infrastructure are in place, the OpenShift Container Platform bootstrap process begins automatically after the RHCOS nodes have rebooted.

   NOTE

   RHCOS nodes do not include a default password for the **core** user. You can access the nodes by running **ssh core@<node>.<cluster_name>.<base_domain>** as a user with access to the SSH private key that is paired to the public key that you specified in your **install_config.yaml** file. OpenShift Container Platform 4 cluster nodes running RHCOS are immutable and rely on Operators to apply cluster changes. Accessing cluster nodes by using SSH is not recommended. However, when investigating installation issues, if the OpenShift Container Platform API is not available, or the kubelet is not properly functioning on a target node, SSH access might be required for debugging or disaster recovery.

4.11.3. 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.

4.11.3.1. Using advanced networking options for PXE and ISO installations

Networking for OpenShift Container Platform nodes uses DHCP by default to gather all necessary configuration settings. To set up static IP addresses or configure special settings, such as bonding, you can do one of the following:

- Pass special kernel parameters when you boot the live installer.
- Use a machine config to copy networking files to the installed system.
- Configure networking from a live installer shell prompt, then copy those settings to the installed system so that they take effect when the installed system first boots.

To configure a PXE or iPXE installation, use one of the following options:

- See the "Advanced RHCOS installation reference" tables.
- Use a machine config to copy networking files to the installed system.

To configure an ISO installation, use the following procedure.

Procedure

1. Boot the ISO installer.
2. From the live system shell prompt, configure networking for the live system using available RHEL tools, such as **nmcli** or **nmtui**.
3. Run the **coreos-installer** command to install the system, adding the **--copy-network** option to copy networking configuration. For example:

```
$ sudo coreos-installer install --copy-network \
    --ignition-url=http://host/worker.ign /dev/sda
```

> **IMPORTANT**
>
> The **--copy-network** option only copies networking configuration found under **/etc/NetworkManager/system-connections**. In particular, it does not copy the system hostname.

4. 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.

4.11.3.2. Disk partitioning

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 you override the default partitioning configuration. During the RHCOS installation, the size of the root file system is increased to use any remaining available space on the target device.

IMPORTANT

The use of a custom partition scheme on your node might result in OpenShift Container Platform not monitoring or alerting on some node partitions. If you override the default partitioning, see Understanding OpenShift File System Monitoring (eviction conditions) for more information about how OpenShift Container Platform monitors your host file systems.

OpenShift Container Platform monitors the following two filesystem identifiers:

- **nodefs**, which is the filesystem that contains **/var/lib/kubelet**
- **imagefs**, which is the filesystem that contains **/var/lib/containers**

For the default partition scheme, **nodefs** and **imagefs** monitor the same root filesystem, /.

To override the default partitioning when installing RHCOS on an OpenShift Container Platform cluster node, you must create separate partitions.

IMPORTANT

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.

Consider a situation where you want to add a separate storage partition for your containers and container images. For example, by mounting **/var/lib/containers** in a separate partition, the kubelet separately monitors **/var/lib/containers** as the **imagefs** directory and the root file system as the **nodefs** directory.

IMPORTANT

If you have resized your disk size to host a larger file system, consider creating a separate **/var/lib/containers** partition. Consider resizing a disk that has an **xfs** format to reduce CPU time issues caused by a high number of allocation groups.

4.11.3.2.1. 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.

IMPORTANT

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

1. 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>
```

2. 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: 4.12.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/<device_name> (1)
    partitions:
    - label: var
      start_mib: <partition_start_offset> (2)
      size_mib: <partition_size> (3)
      number: 5
  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.

(2) 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 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.

> **NOTE**
>
> 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.

3. 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
```

4. 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.

4.11.3.2.2. 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.

> **NOTE**
>
> 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/sda
```

The following example illustrates running the **coreos-installer** in a way that preserves the sixth (6) partition on the disk:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign \
      --save-partindex 6 /dev/sda
```

This example preserves partitions 5 and higher:

```
# coreos-installer install --ignition-url http://10.0.2.2:8080/user.ign
      --save-partindex 5- /dev/sda
```

In the previous examples where partition saving is used, **coreos-installer** recreates the partition immediately.

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
```

4.11.3.3. Identifying Ignition configs

When doing an RHCOS manual installation, there are two types of Ignition configs that you can provide, with different reasons for providing each one:

- **Permanent install Ignition config**: Every manual RHCOS installation needs to pass one of the Ignition config files generated by **openshift-installer**, such as **bootstrap.ign**, **master.ign** and **worker.ign**, to carry out the installation.

IMPORTANT

It is not recommended to modify these 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.

4.11.3.4. Default console configuration

Red Hat Enterprise Linux CoreOS (RHCOS) nodes installed from an OpenShift Container Platform 4.12 boot image use a default console that is meant to accomodate 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.

NOTE

For advanced customization, perform console configuration using the **coreos-installer iso** or **coreos-installer pxe** subcommands, and not kernel arguments.

4.11.3.5. 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

1. Boot the ISO installer.
2. 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/sda
```

(1) The desired secondary console. In this case, the graphical console. Omitting this option will disable the graphical console.

(2) 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 **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.

3. Reboot into the installed system.

> **NOTE**
>
> 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.

4.11.3.6. 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.

4.11.3.7. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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/sda (2)
```

(1) The Ignition config file that is generated from the **openshift-installer** installation program.

(2) When you specify this option, the ISO image automatically runs an installation. Otherwise, the image remains configured for installation, but does not install automatically unless you specify the **coreos.inst.install_dev** kernel argument.

3. 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.

4.11.3.7.1. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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/sda (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. In this case, **/dev/sda**. 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.

> **NOTE**
>
> 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.

3. 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.

4.11.3.7.2. 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.

> **NOTE**
>
> Custom CA certificates affect how Ignition fetches remote resources but they do not affect the certificates installed onto the system.

Procedure

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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
```

IMPORTANT

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.

4.11.3.7.3. 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.

WARNING

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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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

[bond]
miimon=100
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto
```

3. 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
slave-type=bond
```

4. 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
slave-type=bond
```

5. 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.

4.11.3.8. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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/sda \ (2)
    -o rhcos-<version>-custom-initramfs.x86_64.img (3)
```

(1) The Ignition config file that is generated from **openshift-installer**.

(2) When you specify this option, the PXE environment automatically runs an install. Otherwise, the image remains 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.

4.11.3.8.1. 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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.
2. 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/sda \ (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.

4.11.3.8.2. 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.

NOTE

Custom CA certificates affect how Ignition fetches remote resources but they do not affect the certificates installed onto the system.

Procedure

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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
```

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.

IMPORTANT

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.

4.11.3.8.3. 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.

WARNING

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

1. Download the **coreos-installer** binary from the **coreos-installer** image mirror page.

2. 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

[bond]
miimon=100
```

```
mode=active-backup

[ipv4]
method=auto

[ipv6]
method=auto
```

3. 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
slave-type=bond
```

4. 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
slave-type=bond
```

5. 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
```

6. 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.

4.11.3.9. 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.

4.11.3.9.1. 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.

IMPORTANT

When adding networking arguments manually, you must also add the **rd.neednet=1** kernel argument to bring the network up in the initramfs.

The following information provides examples for configuring networking and bonding on your RHCOS nodes for ISO installations. The examples describe how to use the **ip=**, **nameserver=**, and **bond=** kernel arguments.

NOTE

Ordering is important when adding the kernel arguments: **ip=**, **nameserver=**, and then **bond=**.

The networking options are passed to the **dracut** tool during system boot. For more information about the networking options supported by **dracut**, see 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
```

NOTE

When you use DHCP to configure IP addressing for the RHCOS machines, the machines also obtain the DNS server information through DHCP. For DHCP-based deployments, you can define the DNS server address that is used by the RHCOS nodes through your DHCP server configuration.

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.

> **NOTE**
>
> When you configure one or multiple networks, one default gateway is required. If the additional network gateway is different from the primary network gateway, the default gateway must be the primary network gateway.

- 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 network interfaces to a single interface
Optional: You can configure VLANs on bonded interfaces by using the **vlan=** parameter and to use DHCP, for example:

```
ip=bond0.100:dhcp
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

Use the following example to configure the bonded interface with a VLAN and to use a static IP address:

```
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0.100:none
bond=bond0:em1,em2:mode=active-backup
vlan=bond0.100:bond0
```

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**).

NOTE

Teaming is planned to be deprecated when RHCOS switches to an upcoming version of RHEL. For more information, see this Red Hat Knowledgebase Article .

Use the following example to configure a network team:

```
team=team0:em1,em2
ip=team0:dhcp
```

4.11.3.9.2. **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 4.12. coreos-installer subcommands, command-line options, and arguments

coreos-installer install subcommand	
Subcommand	*Description*
$ coreos-installer install <options> <device>	Embed an Ignition config in an ISO image.
coreos-installer install subcommand options	
Option	*Description*
-u, **--image-url <url>**	Specify the image URL manually.

-f, --image-file <path>	Specify a local image file manually. Used for debugging.
-i, --ignition-file <path>	Embed an Ignition config from a file.
-I, --ignition-url <URL>	Embed an Ignition config from a URL.
--ignition-hash <digest>	Digest **type-value** of the Ignition config.
-p, --platform <name>	Override the Ignition platform ID for the installed system.
--console <spec>	Set the kernel and bootloader console for the installed system. For more information about the format of **<spec>**, see the Linux kernel serial console documentation.
--append-karg <arg>...	Append a default kernel argument to the installed system.
--delete-karg <arg>...	Delete a default kernel argument from the installed system.
-n, --copy-network	Copy the network configuration from the install environment. IMPORTANT The **--copy-network** option only copies networking configuration found under **/etc/NetworkManager/system-connections**. In particular, it does not copy the system hostname.
--network-dir <path>	For use with **-n**. Default is **/etc/NetworkManager/system-connections/**.
--save-partlabel <lx>..	Save partitions with this label glob.
--save-partindex <id>...	Save partitions with this number or range.
--insecure	Skip RHCOS image signature verification.
--insecure-ignition	Allow Ignition URL without HTTPS or hash.
--architecture <name>	Target CPU architecture. Valid values are **x86_64** and **aarch64**.

--preserve-on-error	Do not clear partition table on error.
-h, --help	Print help information.
coreos-installer install subcommand argument	
Argument	*Description*
<device>	The destination device.
coreos-installer ISO subcommands	
Subcommand	*Description*
$ coreos-installer iso customize <options> <ISO_image>	Customize a RHCOS live ISO image.
coreos-installer iso reset <options> <ISO_image>	Restore a RHCOS live ISO image to default settings.
coreos-installer iso ignition remove <options> <ISO_image>	Remove the embedded Ignition config from an ISO image.
coreos-installer ISO customize subcommand options	
Option	*Description*
--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and 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	
Subcommand	*Description*
Note that not all of these options are accepted by all subcommands.	
coreos-installer pxe customize <options> <path>	Customize a RHCOS live PXE boot config.
coreos-installer pxe ignition wrap <options>	Wrap an Ignition config in an image.
coreos-installer pxe ignition unwrap <options> <image_name>	Show the wrapped Ignition config in an image.
coreos-installer PXE customize subcommand options	
Option	*Description*
Note that not all of these options are accepted by all subcommands.	

--dest-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the destination system.
--dest-console <spec>	Specify the kernel and 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.
post-install <path>	Run the specified script after installation.
--installer-config <path>	Apply the specified installer configuration file.
--live-ignition <path>	Merge the specified Ignition config file into a new configuration fragment for the live environment.
-o, --output <path>	Write the initramfs to a new output file. **NOTE** This option is required for PXE environments.
-h, --help	Print help information.

4.11.3.9.3. **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 4.13. coreos.inst boot options

Argument	Description
coreos.inst.install_dev	Required. The block device on the system to install to. It is recommended to use the full path, such as **/dev/sda**, although **sda** is allowed.
coreos.inst.ignition_url	Optional: The URL of the Ignition config to embed into the installed system. If no URL is specified, no Ignition config is embedded. Only HTTP and HTTPS protocols are supported.
coreos.inst.save_partlabel	Optional: Comma-separated labels of partitions to preserve during the install. Glob-style wildcards are permitted. The specified partitions do not need to exist.
coreos.inst.save_partindex	Optional: Comma-separated indexes of partitions to preserve during the install. Ranges **m-n** are permitted, and either **m** or **n** can be omitted. The specified partitions do not need to exist.
coreos.inst.insecure	Optional: Permits the OS image that is specified by **coreos.inst.image_url** to be unsigned.
coreos.inst.image_url	Optional: Download and install the specified RHCOS image. • This argument should not be used in production environments and is intended for debugging purposes only. • While this argument can be used to install a version of RHCOS that does not match the live media, it is recommended that you instead use the media that matches the version you want to install. • If you are using **coreos.inst.image_url**, you must also use **coreos.inst.insecure**. This is because the bare-metal media are not GPG-signed for OpenShift Container Platform. • Only HTTP and HTTPS protocols are supported.
coreos.inst.skip_reboot	Optional: The system will not reboot after installing. After the install finishes, you will receive a prompt that allows you to inspect what is happening during installation. This argument should not be used in production environments and is intended for debugging purposes only.

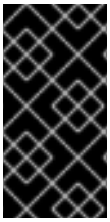
Argument	Description
coreos.inst.platform_id	Optional: The Ignition platform ID of the platform the RHCOS image is being installed on. Default is **metal**. This option determines whether or not to request an Ignition config from the cloud provider, such as VMware. For example: **coreos.inst.platform_id=vmware**.
ignition.config.url	Optional: The URL of the Ignition config for the live boot. For example, this can be used to customize how **coreos-installer** is invoked, or to run code before or after the installation. This is different from **coreos.inst.ignition_url**, which is the Ignition config for the installed system.

4.11.4. 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.



IMPORTANT

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.

NOTE

OpenShift Container Platform does not support enabling multipathing as a day-2 activity on nodes that have been upgraded from 4.6 or earlier.

Prerequisites

- You have created the Ignition config files for your cluster.
- You have reviewed *Installing RHCOS and starting the OpenShift Container Platform bootstrap process.*

Procedure

1. To enable multipath and start the **multipathd** daemon, run the following command on the installation host:

```
$ 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.

2. 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)
--ignition-url=http://host/worker.ign \
--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)
--ignition-url=http://host/worker.ign \
--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".

3. Reboot into the installed system.

4. 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.

4.11.4.1. Enabling multipathing on secondary disks

RHCOS also supports multipathing on a secondary disk. Instead of kernel arguments, you use Ignition to enable multipathing for the secondary disk at installation time.

Prerequisites

- You have read the section *Disk partitioning*.
- You have read *Enabling multipathing with kernel arguments on RHCOS* .
- You have installed the Butane utility.

Procedure

1. Create a Butane config with information similar to the following:

   Example multipath-config.bu

```
variant: openshift
version: 4.12.0
systemd:
  units:
    - name: mpath-configure.service
      enabled: true
      contents: |
        [Unit]
        Description=Configure Multipath on Secondary Disk
        ConditionFirstBoot=true
        ConditionPathExists=!/etc/multipath.conf
        Before=multipathd.service (1)
        DefaultDependencies=no

        [Service]
        Type=oneshot
        ExecStart=/usr/sbin/mpathconf --enable (2)

        [Install]
        WantedBy=multi-user.target
    - name: mpath-var-lib-container.service
      enabled: true
      contents: |
        [Unit]
        Description=Set Up Multipath On /var/lib/containers
        ConditionFirstBoot=true (3)
        Requires=dev-mapper-mpatha.device
        After=dev-mapper-mpatha.device
        After=ostree-remount.service
```

```
      Before=kubelet.service
      DefaultDependencies=no

      [Service] (4)
      Type=oneshot
      ExecStart=/usr/sbin/mkfs.xfs -L containers -m reflink=1 /dev/mapper/mpatha
      ExecStart=/usr/bin/mkdir -p /var/lib/containers

      [Install]
      WantedBy=multi-user.target
  - name: var-lib-containers.mount
    enabled: true
    contents: |
      [Unit]
      Description=Mount /var/lib/containers
      After=mpath-var-lib-containers.service
      Before=kubelet.service (5)

      [Mount] (6)
      What=/dev/disk/by-label/dm-mpath-containers
      Where=/var/lib/containers
      Type=xfs

      [Install]
      WantedBy=multi-user.target
```

1. The configuration must be set before launching the multipath daemon.
2. Starts the **mpathconf** utility.
3. This field must be set to the value **true**.
4. Creates the filesystem and directory **/var/lib/containers**.
5. The device must be mounted before starting any nodes.
6. Mounts the device to the **/var/lib/containers** mount point. This location cannot be a symlink.

2. Create the Ignition configuration by running the following command:

```
$ butane --pretty --strict multipath-config.bu > multipath-config.ign
```

3. Continue with the rest of the first boot RHCOS installation process.

IMPORTANT

Do not add the **rd.multipath** or **root** kernel arguments on the command-line during installation unless the primary disk is also multipathed.

4.12. 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.

Procedure

1. Monitor the bootstrap process:

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete \ (1)
    --log-level=info (2)
```

(1) For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

(2) To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.25.0 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

The command succeeds when the Kubernetes API server signals that it has been bootstrapped on the control plane machines.

2. After the bootstrap process is complete, remove the bootstrap machine from the load balancer.

IMPORTANT

You must remove the bootstrap machine from the load balancer at this point. You can also remove or reformat the bootstrap machine itself.

Additional resources

- See Monitoring installation progress for more information about monitoring the installation logs and retrieving diagnostic data if installation issues arise.

4.13. LOGGING IN TO THE CLUSTER BY USING THE CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The

kubeconfig file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the **oc** CLI.

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
```

(1) For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration:

```
$ oc whoami
```

Example output

```
system:admin
```

4.14. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS   ROLES    AGE  VERSION
master-0  Ready    master   63m  v1.25.0
master-1  Ready    master   63m  v1.25.0
master-2  Ready    master   64m  v1.25.0
```

The output lists all of the machines that you created.

NOTE

The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> (1)
```

(1) **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

Example output

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> (1)
```

(1) **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.25.0
```

```
master-1  Ready    master  73m  v1.25.0
master-2  Ready    master  74m  v1.25.0
worker-0  Ready    worker  11m  v1.25.0
worker-1  Ready    worker  11m  v1.25.0
```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

Additional information

- For more information on CSRs, see Certificate Signing Requests.

4.15. INITIAL OPERATOR CONFIGURATION

After the control plane initializes, you must immediately configure some Operators so that they all become available.

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False      19m
baremetal                                  4.12.0    True        False         False      37m
cloud-credential                           4.12.0    True        False         False      40m
cluster-autoscaler                         4.12.0    True        False         False      37m
config-operator                            4.12.0    True        False         False      38m
console                                    4.12.0    True        False         False      26m
csi-snapshot-controller                    4.12.0    True        False         False      37m
dns                                        4.12.0    True        False         False      37m
etcd                                       4.12.0    True        False         False      36m
image-registry                             4.12.0    True        False         False      31m
ingress                                    4.12.0    True        False         False      30m
insights                                   4.12.0    True        False         False      31m
kube-apiserver                             4.12.0    True        False         False      26m
kube-controller-manager                    4.12.0    True        False         False      36m
kube-scheduler                             4.12.0    True        False         False      36m
kube-storage-version-migrator              4.12.0    True        False         False      37m
machine-api                                4.12.0    True        False         False      29m
machine-approver                           4.12.0    True        False         False      37m
machine-config                             4.12.0    True        False         False      36m
marketplace                                4.12.0    True        False         False      37m
monitoring                                 4.12.0    True        False         False      29m
```

```
network                                    4.12.0   True        False        False      38m
node-tuning                                4.12.0   True        False        False      37m
openshift-apiserver                        4.12.0   True        False        False      32m
openshift-controller-manager               4.12.0   True        False        False      30m
openshift-samples                          4.12.0   True        False        False      32m
operator-lifecycle-manager                 4.12.0   True        False        False      37m
operator-lifecycle-manager-catalog         4.12.0   True        False        False      37m
operator-lifecycle-manager-packageserver   4.12.0   True        False        False      32m
service-ca                                 4.12.0   True        False        False      38m
storage                                    4.12.0   True        False        False      37m
```

2. 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.

4.15.1. Disabling the default OperatorHub catalog sources

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

4.15.2. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

4.15.2.1. Changing the image registry's management state

To start the image registry, you must change the Image Registry Operator configuration's **managementState** from **Removed** to **Managed**.

Procedure

- Change **managementState** Image Registry Operator configuration from **Removed** to **Managed**. For example:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
```

4.15.2.2. Configuring registry storage for bare metal and other manual installations

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have a cluster that uses manually-provisioned Red Hat Enterprise Linux CoreOS (RHCOS) nodes, such as bare metal.
- You have provisioned persistent storage for your cluster, such as Red Hat OpenShift Data Foundation.

> **IMPORTANT**
>
> OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. **ReadWriteOnce** access also requires that the registry uses the **Recreate** rollout strategy. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

> **NOTE**
>
> When you use shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

Example output

```
No resources found in openshift-image-registry namespace
```

NOTE

If you do have a registry pod in your output, you do not need to continue with this procedure.

3. Check the registry configuration:

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Example output

```
storage:
  pvc:
    claim:
```

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

```
$ oc get clusteroperator image-registry
```

Example output

```
NAME            VERSION            AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.12               True        False         False      6h50m
```

5. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

```
$ oc edit configs.imageregistry/cluster
```

Then, change the line

```
managementState: Removed
```

to

```
managementState: Managed
```

4.15.2.3. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

- To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

WARNING

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

4.15.2.4. Configuring block registry storage for bare metal

To allow the image registry to use block storage types during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.

IMPORTANT

Block storage volumes, or block persistent volumes, are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

If you choose to use a block storage volume with the image registry, you must use a filesystem persistent volume claim (PVC).

Procedure

1. Enter the following command to set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy, and runs with only one (**1**) replica:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.

   a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (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.

b. Enter the following command to create the **PersistentVolumeClaim** object from the file:

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. Enter the following command to edit the registry configuration so that it references the correct PVC:

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

Example output

```
storage:
  pvc:
    claim: (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.

4.16. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE

After you complete the Operator configuration, you can finish installing the cluster on infrastructure that you provide.

Prerequisites

- Your control plane has initialized.
- You have completed the initial Operator configuration.

Procedure

1. Confirm that all the cluster components are online with the following command:

```
$ watch -n5 oc get clusteroperators
```

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    True        False         False         19m
baremetal                                  4.12.0    True        False         False         37m
cloud-credential                           4.12.0    True        False         False         40m
cluster-autoscaler                         4.12.0    True        False         False         37m
config-operator                            4.12.0    True        False         False         38m
console                                    4.12.0    True        False         False         26m
csi-snapshot-controller                    4.12.0    True        False         False         37m
dns                                        4.12.0    True        False         False         37m
etcd                                       4.12.0    True        False         False         36m
image-registry                             4.12.0    True        False         False         31m
ingress                                    4.12.0    True        False         False         30m
insights                                   4.12.0    True        False         False         31m
kube-apiserver                             4.12.0    True        False         False         26m
kube-controller-manager                    4.12.0    True        False         False         36m
kube-scheduler                             4.12.0    True        False         False         36m
kube-storage-version-migrator              4.12.0    True        False         False         37m
machine-api                                4.12.0    True        False         False         29m
machine-approver                           4.12.0    True        False         False         37m
machine-config                             4.12.0    True        False         False         36m
marketplace                                4.12.0    True        False         False         37m
monitoring                                 4.12.0    True        False         False         29m
network                                    4.12.0    True        False         False         38m
node-tuning                                4.12.0    True        False         False         37m
openshift-apiserver                        4.12.0    True        False         False         32m
openshift-controller-manager               4.12.0    True        False         False         30m
openshift-samples                          4.12.0    True        False         False         32m
operator-lifecycle-manager                 4.12.0    True        False         False         37m
operator-lifecycle-manager-catalog         4.12.0    True        False         False         37m
operator-lifecycle-manager-packageserver   4.12.0    True        False         False         32m
service-ca                                 4.12.0    True        False         False         38m
storage                                    4.12.0    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.

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Confirm that the Kubernetes API server is communicating with the pods.

   a. To view a list of all pods, use the following command:

   ```
   $ oc get pods --all-namespaces
   ```

   Example output

   ```
   NAMESPACE                    NAME                                    READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator     openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running     1          9m
   openshift-apiserver              apiserver-67b9g                          1/1     Running     0
   3m
   openshift-apiserver              apiserver-ljcmx                          1/1     Running     0
   1m
   openshift-apiserver              apiserver-z25h4                          1/1     Running     0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
   Running     0          5m
   ...
   ```

   b. View the logs for a pod that is listed in the output of the previous command by using the following command:

   ```
   $ oc logs <pod_name> -n <namespace> (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.

3. For an installation with Fibre Channel Protocol (FCP), additional steps are required to enable multipathing. Do not enable multipathing during installation.
   See "Enabling multipathing with kernel arguments on RHCOS" in the *Post-installation machine configuration tasks* documentation for more information.

4. Register your cluster on the Cluster registration page.

4.17. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM

In OpenShift Container Platform 4.12, 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 Hybrid Cloud Console.

After you confirm that your OpenShift Cluster Manager Hybrid Cloud Console 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

4.18. NEXT STEPS

- Validating an installation.
- Customize your cluster.
- Configure image streams for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to use Operator Lifecycle Manager (OLM) on restricted networks .
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.
- Remote health reporting
- If necessary, see Registering your disconnected cluster