OpenShift Container Platform 4.12

Installing on IBM Z and IBM LinuxONE

Installing OpenShift Container Platform on IBM Z and IBM LinuxONE

Last Updated: 2024-09-11

OpenShift Container Platform 4.12 Installing on IBM Z and IBM LinuxONE

Installing OpenShift Container Platform on IBM Z and IBM LinuxONE

Abstract

This document describes how to install OpenShift Container Platform on IBM Z and IBM LinuxONE.

Table of Contents

CHAPTER 1. PREPARING TO INSTALL ON IBM Z® AND {LINUXONEPRODUCTNAME} 8
1.1. PREREQUISITES 8
1.2. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON IBM Z OR IBM(R) LINUXONE 8
1.2.1. User-provisioned infrastructure installation of OpenShift Container Platform on IBM Z 9

CHAPTER 2. INSTALLING A CLUSTER WITH Z/VM ON IBM Z AND IBM(R) LINUXONE 10
2.1. PREREQUISITES 10
2.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 10
2.3. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 11
2.3.1. Required machines for cluster installation 11
2.3.2. Minimum resource requirements for cluster installation 11
2.3.3. Minimum IBM Z system environment 12
Hardware requirements 12
Operating system requirements 13
IBM Z network connectivity requirements 13
Disk storage for the z/VM guest virtual machines 13
Storage / Main Memory 13
2.3.4. Preferred IBM Z system environment 13
Hardware requirements 13
Operating system requirements 14
IBM Z network connectivity requirements 14
Disk storage for the z/VM guest virtual machines 14
Storage / Main Memory 14
2.3.5. Certificate signing requests management 14
2.3.6. Networking requirements for user-provisioned infrastructure 15
2.3.6.1. Network connectivity requirements 15
NTP configuration for user-provisioned infrastructure 16
2.3.7. User-provisioned DNS requirements 16
2.3.7.1. Example DNS configuration for user-provisioned clusters 18
2.3.8. Load balancing requirements for user-provisioned infrastructure 20
2.3.8.1. 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 25
2.6. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 28
2.7. OBTAINING THE INSTALLATION PROGRAM 29
2.8. INSTALLING THE OPENSHIFT CLI BY DOWNLOADING THE BINARY 30
Installing the OpenShift CLI on Linux 30
Installing the OpenShift CLI on Windows 31
Installing the OpenShift CLI on macOS 31
2.9. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 32
2.9.1. Sample install-config.yaml file for IBM Z 33
2.9.2. Configuring the cluster-wide proxy during installation 35
2.9.3. Configuring a three-node cluster 37
2.10. CLUSTER NETWORK OPERATOR CONFIGURATION 38
2.10.1. Cluster Network Operator configuration object 38
defaultNetwork object configuration 39
Configuration for the OpenShift SDN network plugin 40
Configuration for the OVN-Kubernetes network plugin 41
kubeProxyConfig object configuration 45
2.11. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES 46

2.12. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 48
2.12.1. Advanced RHCOS installation reference 51
2.12.1.1. Networking and bonding options for ISO installations 51
Configuring DHCP or static IP addresses 51
Configuring an IP address without a static hostname 52
Specifying multiple network interfaces 52
Configuring default gateway and route 52
Disabling DHCP on a single interface 53
Combining DHCP and static IP configurations 53
Configuring VLANs on individual interfaces 53
Providing multiple DNS servers 53
Bonding multiple network interfaces to a single interface 53
Bonding multiple network interfaces to a single interface 54
Using network teaming 54
2.13. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 54
2.14. LOGGING IN TO THE CLUSTER BY USING THE CLI 55
2.15. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 56
2.16. INITIAL OPERATOR CONFIGURATION 59
2.16.1. Image registry storage configuration 60
2.16.1.1. Configuring registry storage for IBM Z 60
2.16.1.2. Configuring storage for the image registry in non-production clusters 61
2.17. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 62
2.18. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM 65
2.19. NEXT STEPS 65

CHAPTER 3. INSTALLING A CLUSTER WITH Z/VM ON IBM Z AND IBM(R) LINUXONE IN A RESTRICTED NETWORK . . . 66
3.1. PREREQUISITES 66
3.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS 66
3.2.1. Additional limits 67
3.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 67
3.4. REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 68
3.4.1. Required machines for cluster installation 68
3.4.2. Minimum resource requirements for cluster installation 68
3.4.3. Minimum IBM Z system environment 69
Hardware requirements 69
Operating system requirements 70
IBM Z network connectivity requirements 70
Disk storage for the z/VM guest virtual machines 70
Storage / Main Memory 70
3.4.4. Preferred IBM Z system environment 70
Hardware requirements 70
Operating system requirements 70
IBM Z network connectivity requirements 71
Disk storage for the z/VM guest virtual machines 71
Storage / Main Memory 71
3.4.5. Certificate signing requests management 71
3.4.6. Networking requirements for user-provisioned infrastructure 72
3.4.6.1. Setting the cluster node hostnames through DHCP 72
3.4.6.2. Network connectivity requirements 72
NTP configuration for user-provisioned infrastructure 73
3.4.7. User-provisioned DNS requirements 74

3.4.7.1. Example DNS configuration for user-provisioned clusters 76
3.4.8. Load balancing requirements for user-provisioned infrastructure 78
3.4.8.1. Example load balancer configuration for user-provisioned clusters 80
3.5. PREPARING THE USER-PROVISIONED INFRASTRUCTURE 81
3.6. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE 83
3.7. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 85
3.8. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 87
3.8.1. Sample install-config.yaml file for IBM Z 88
3.8.2. Configuring the cluster-wide proxy during installation 90
3.8.3. Configuring a three-node cluster 92
3.9. CLUSTER NETWORK OPERATOR CONFIGURATION 93
3.9.1. Cluster Network Operator configuration object 94
defaultNetwork object configuration 94
Configuration for the OpenShift SDN network plugin 95
Configuration for the OVN-Kubernetes network plugin 96
kubeProxyConfig object configuration 100
3.10. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES 101
3.11. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 103
3.11.1. Advanced RHCOS installation reference 106
3.11.1.1. Networking and bonding options for ISO installations 106
Configuring DHCP or static IP addresses 107
Configuring an IP address without a static hostname 107
Specifying multiple network interfaces 107
Configuring default gateway and route 108
Disabling DHCP on a single interface 108
Combining DHCP and static IP configurations 108
Configuring VLANs on individual interfaces 108
Providing multiple DNS servers 108
Bonding multiple network interfaces to a single interface 109
Bonding multiple network interfaces to a single interface 109
Using network teaming 109
3.12. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 110
3.13. LOGGING IN TO THE CLUSTER BY USING THE CLI 110
3.14. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 111
3.15. INITIAL OPERATOR CONFIGURATION 114
3.15.1. Disabling the default OperatorHub catalog sources 115
3.15.2. Image registry storage configuration 115
3.15.2.1. Configuring registry storage for IBM Z 115
3.15.2.2. Configuring storage for the image registry in non-production clusters 117
3.16. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 118
3.17. NEXT STEPS 120

CHAPTER 4. INSTALLING A CLUSTER WITH RHEL KVM ON IBM Z AND IBM(R) LINUXONE 121
4.1. PREREQUISITES 121
4.2. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 121
4.3. MACHINE REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 122
4.3.1. Required machines 122
4.3.2. Network connectivity requirements 122
4.3.3. IBM Z network connectivity requirements 123
4.3.4. Host machine resource requirements 123
4.3.5. Minimum IBM Z system environment 123
Hardware requirements 123

Operating system requirements 124
4.3.6. Minimum resource requirements 124
4.3.7. Preferred IBM Z system environment 124
Hardware requirements 124
Operating system requirements 124
4.3.8. Preferred resource requirements 125
4.3.9. Certificate signing requests management 125
4.3.10. Networking requirements for user-provisioned infrastructure 125
4.3.10.1. Setting the cluster node hostnames through DHCP 126
4.3.10.2. Network connectivity requirements 126
NTP configuration for user-provisioned infrastructure 127
4.3.11. User-provisioned DNS requirements 128
4.3.11.1. Example DNS configuration for user-provisioned clusters 130
4.3.12. Load balancing requirements for user-provisioned infrastructure 132
4.3.12.1. Example load balancer configuration for user-provisioned clusters 134
4.4. PREPARING THE USER-PROVISIONED INFRASTRUCTURE 135
4.5. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE 137
4.6. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 140
4.7. OBTAINING THE INSTALLATION PROGRAM 142
4.8. INSTALLING THE OPENSHIFT CLI BY DOWNLOADING THE BINARY 142
Installing the OpenShift CLI on Linux 143
Installing the OpenShift CLI on Windows 143
Installing the OpenShift CLI on macOS 144
4.9. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 144
4.9.1. Sample install-config.yaml file for IBM Z 145
4.9.2. Configuring the cluster-wide proxy during installation 148
4.9.3. Configuring a three-node cluster 149
4.10. CLUSTER NETWORK OPERATOR CONFIGURATION 150
4.10.1. Cluster Network Operator configuration object 151
defaultNetwork object configuration 152
Configuration for the OpenShift SDN network plugin 152
Configuration for the OVN-Kubernetes network plugin 153
kubeProxyConfig object configuration 157
4.11. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES 158
4.12. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 160
4.12.1. Installing RHCOS using IBM Secure Execution 160
4.12.2. Fast-track installation by using a prepackaged QCOW2 disk image 162
4.12.3. Full installation on a new QCOW2 disk image 163
4.12.4. Advanced RHCOS installation reference 165
4.12.4.1. Networking options for ISO installations 165
Configuring DHCP or static IP addresses 165
Configuring an IP address without a static hostname 166
Specifying multiple network interfaces 166
Configuring default gateway and route 166
Disabling DHCP on a single interface 166
Combining DHCP and static IP configurations 167
Configuring VLANs on individual interfaces 167
Providing multiple DNS servers 167
4.13. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 167
4.14. LOGGING IN TO THE CLUSTER BY USING THE CLI 168
4.15. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 169
4.16. INITIAL OPERATOR CONFIGURATION 171

4.16.1. Image registry storage configuration 172
4.16.1.1. Configuring registry storage for IBM Z 172
4.16.1.2. Configuring storage for the image registry in non-production clusters 174
4.17. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 175
4.18. TELEMETRY ACCESS FOR OPENSHIFT CONTAINER PLATFORM 177
4.19. NEXT STEPS 177

CHAPTER 5. INSTALLING A CLUSTER WITH RHEL KVM ON IBM Z AND IBM(R) LINUXONE IN A RESTRICTED NETWORK 178
5.1. PREREQUISITES 178
5.2. ABOUT INSTALLATIONS IN RESTRICTED NETWORKS 178
5.2.1. Additional limits 179
5.3. INTERNET ACCESS FOR OPENSHIFT CONTAINER PLATFORM 179
5.4. MACHINE REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE 180
5.4.1. Required machines 180
5.4.2. Network connectivity requirements 180
5.4.3. IBM Z network connectivity requirements 180
5.4.4. Host machine resource requirements 181
5.4.5. Minimum IBM Z system environment 181
Hardware requirements 181
Operating system requirements 181
5.4.6. Minimum resource requirements 182
5.4.7. Preferred IBM Z system environment 182
Hardware requirements 182
Operating system requirements 182
5.4.8. Preferred resource requirements 182
5.4.9. Certificate signing requests management 183
5.4.10. Networking requirements for user-provisioned infrastructure 183
5.4.10.1. Setting the cluster node hostnames through DHCP 184
5.4.10.2. Network connectivity requirements 184
NTP configuration for user-provisioned infrastructure 185
5.4.11. User-provisioned DNS requirements 185
5.4.11.1. Example DNS configuration for user-provisioned clusters 187
5.4.12. Load balancing requirements for user-provisioned infrastructure 189
5.4.12.1. Example load balancer configuration for user-provisioned clusters 191
5.5. PREPARING THE USER-PROVISIONED INFRASTRUCTURE 193
5.6. VALIDATING DNS RESOLUTION FOR USER-PROVISIONED INFRASTRUCTURE 195
5.7. GENERATING A KEY PAIR FOR CLUSTER NODE SSH ACCESS 197
5.8. MANUALLY CREATING THE INSTALLATION CONFIGURATION FILE 199
5.8.1. Sample install-config.yaml file for IBM Z 199
5.8.2. Configuring the cluster-wide proxy during installation 202
5.8.3. Configuring a three-node cluster 204
5.9. CLUSTER NETWORK OPERATOR CONFIGURATION 205
5.9.1. Cluster Network Operator configuration object 206
defaultNetwork object configuration 206
Configuration for the OpenShift SDN network plugin 207
Configuration for the OVN-Kubernetes network plugin 208
kubeProxyConfig object configuration 212
5.10. CREATING THE KUBERNETES MANIFEST AND IGNITION CONFIG FILES 213
5.11. INSTALLING RHCOS AND STARTING THE OPENSHIFT CONTAINER PLATFORM BOOTSTRAP PROCESS 215
5.11.1. Installing RHCOS using IBM Secure Execution 215
5.11.2. Fast-track installation by using a prepackaged QCOW2 disk image 217

5.11.3. Full installation on a new QCOW2 disk image 218
5.11.4. Advanced RHCOS installation reference 220
5.11.4.1. Networking options for ISO installations 220
Configuring DHCP or static IP addresses 220
Configuring an IP address without a static hostname 221
Specifying multiple network interfaces 221
Configuring default gateway and route 221
Disabling DHCP on a single interface 221
Combining DHCP and static IP configurations 222
Configuring VLANs on individual interfaces 222
Providing multiple DNS servers 222
5.12. WAITING FOR THE BOOTSTRAP PROCESS TO COMPLETE 222
5.13. LOGGING IN TO THE CLUSTER BY USING THE CLI 223
5.14. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES 224
5.15. INITIAL OPERATOR CONFIGURATION 226
5.15.1. Disabling the default OperatorHub catalog sources 227
5.15.2. Image registry storage configuration 228
5.15.2.1. Configuring registry storage for IBM Z 228
5.15.2.2. Configuring storage for the image registry in non-production clusters 230
5.16. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE 230
5.17. NEXT STEPS 233

CHAPTER 6. INSTALLATION CONFIGURATION PARAMETERS FOR IBM Z® AND IBM(R) LINUXONE . . 234
6.1. INSTALLATION CONFIGURATION PARAMETERS 234
6.1.1. Required configuration parameters 234
6.1.2. Network configuration parameters 235
6.1.3. Optional configuration parameters 237

CHAPTER 1. PREPARING TO INSTALL ON IBM Z® AND {LINUXONEPRODUCTNAME}

1.1. PREREQUISITES

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- Before you begin the installation process, you must clean the installation directory. This ensures that the required installation files are created and updated during the installation process.
- You provisioned persistent storage using OpenShift Data Foundation or other supported storage protocols for your cluster. To deploy a private image registry, you must set up persistent storage with **ReadWriteMany** access.
- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

> **NOTE**
>
> While this document refers only to IBM Z, all information in it also applies to IBM® LinuxONE.

1.2. CHOOSING A METHOD TO INSTALL OPENSHIFT CONTAINER PLATFORM ON IBM Z OR IBM(R) LINUXONE

You can install OpenShift Container Platform with the Assisted Installer. This method requires no setup for the installer, and is ideal for connected environments like IBM Z. See Installing an on-premise cluster using the Assisted Installer for additional details.

> **NOTE**
>
> Installing OpenShift Container Platform with the Assisted Installer on IBM Z is supported only with RHEL KVM installations.

You can also install OpenShift Container Platform on infrastructure that you provide. If you do not use infrastructure that the installation program provisions, you must manage and maintain the cluster resources yourself.

See the Installation process for more information about Assisted Installer and user-provisioned installation processes.

> **IMPORTANT**
>
> The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the IBM Z platform and the installation process of OpenShift Container Platform. Use the user-provisioned infrastructure installation instructions as a guide; you are free to create the required resources through other methods.

1.2.1. User-provisioned infrastructure installation of OpenShift Container Platform on IBM Z

User-provisioned infrastructure requires the user to provision all resources required by OpenShift Container Platform.

- Installing a cluster with z/VM on IBM Z and IBM® LinuxONE: You can install OpenShift Container Platform with z/VM on IBM Z or IBM® LinuxONE infrastructure that you provision.
- Installing a cluster with z/VM on IBM Z and IBM® LinuxONE in a restricted network: You can install OpenShift Container Platform with z/VM on IBM Z or IBM® LinuxONE infrastructure that you provision in a restricted or disconnected network, by using an internal mirror of the installation release content. You can use this method to install a cluster that does not require an active internet connection to obtain the software components. You can also use this installation method to ensure that your clusters only use container images that satisfy your organizational controls on external content.
- Installing a cluster with RHEL KVM on IBM Z and IBM® LinuxONE: You can install OpenShift Container Platform with KVM on IBM Z or IBM® LinuxONE infrastructure that you provision.
- Installing a cluster with RHEL KVM on IBM Z and IBM® LinuxONE in a restricted network: You can install OpenShift Container Platform with RHEL KVM on IBM Z or IBM® LinuxONE infrastructure that you provision in a restricted or disconnected network, by using an internal mirror of the installation release content. You can use this method to install a cluster that does not require an active internet connection to obtain the software components. 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 CLUSTER WITH Z/VM ON IBM Z AND IBM(R) LINUXONE

In OpenShift Container Platform version 4.12, you can install a cluster on IBM Z or IBM® LinuxONE infrastructure that you provision.

NOTE

While this document refers only to IBM Z, all information in it also applies to IBM® LinuxONE.

IMPORTANT

Additional considerations exist for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you install an OpenShift Container Platform cluster.

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.
- Before you begin the installation process, you must clean the installation directory. This ensures that the required installation files are created and updated during the installation process.
- You provisioned persistent storage using OpenShift Data Foundation or other supported storage protocols for your cluster. To deploy a private image registry, you must set up persistent storage with **ReadWriteMany** access.
- 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.

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.

IMPORTANT

To improve high availability of your cluster, distribute the control plane machines over different z/VM instances on at least two physical machines.

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

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

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	vCPU [1]	Virtual RAM	Storage	IOPS
Bootstrap	RHCOS	4	16 GB	100 GB	N/A
Control plane	RHCOS	4	16 GB	100 GB	N/A
Compute	RHCOS	2	8 GB	100 GB	N/A

1. One physical core (IFL) provides two logical cores (threads) when SMT-2 is enabled. The hypervisor can provide two or more vCPUs.

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. Minimum IBM Z system environment

You can install OpenShift Container Platform version 4.12 on the following IBM hardware:

- IBM z16 (all models), IBM z15 (all models), IBM z14 (all models), IBM z13, and IBM z13s
- IBM® LinuxONE Emperor 4, IBM® LinuxONE III (all models), IBM® LinuxONE Emperor II, IBM® LinuxONE Rockhopper II, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper

NOTE

Support for RHCOS functionality for IBM z13 all models, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper is deprecated. These hardware models remain fully supported in OpenShift Container Platform 4.12. However, Red Hat recommends that you use later hardware models.

Hardware requirements

- The equivalent of six Integrated Facilities for Linux (IFL), which are SMT2 enabled, for each cluster.
- At least one network connection to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.

NOTE

You can use dedicated or shared IFLs to assign sufficient compute resources. Resource sharing is one of the key strengths of IBM Z. However, you must adjust capacity correctly on each hypervisor layer and ensure sufficient resources for every OpenShift Container Platform cluster.

IMPORTANT

Since the overall performance of the cluster can be impacted, the LPARs that are used to set up the OpenShift Container Platform clusters must provide sufficient compute capacity. In this context, LPAR weight management, entitlements, and CPU shares on the hypervisor level play an important role.

Operating system requirements

- One instance of z/VM 7.2 or later

On your z/VM instance, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines
- Two guest virtual machines for OpenShift Container Platform compute machines
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

IBM Z network connectivity requirements
To install on IBM Z under z/VM, you require a single z/VM virtual NIC in layer 2 mode. You also need:

- A direct-attached OSA or RoCE network adapter
- A z/VM VSwitch set up. For a preferred setup, use OSA link aggregation.

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

2.3.4. Preferred IBM Z system environment

Hardware requirements

- Three LPARS that each have the equivalent of six IFLs, which are SMT2 enabled, for each cluster.
- Two network connections to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.
- HiperSockets, which are attached to a node either directly as a device or by bridging with one z/VM VSWITCH to be transparent to the z/VM guest. To directly connect HiperSockets to a node, you must set up a gateway to the external network via a RHEL 8 guest to bridge to the

HiperSockets network.

Operating system requirements

- Two or three instances of z/VM 7.2 or later for high availability

On your z/VM instances, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines, one per z/VM instance.
- At least six guest virtual machines for OpenShift Container Platform compute machines, distributed across the z/VM instances.
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine.
- To ensure the availability of integral components in an overcommitted environment, increase the priority of the control plane by using the CP command **SET SHARE**. Do the same for infrastructure nodes, if they exist. See SET SHARE in IBM Documentation.

IBM Z network connectivity requirements
To install on IBM Z under z/VM, you require a single z/VM virtual NIC in layer 2 mode. You also need:

- A direct-attached OSA or RoCE network adapter
- A z/VM VSwitch set up. For a preferred setup, use OSA link aggregation.

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV and High Performance FICON (zHPF) to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

2.3.5. 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 Bridging a HiperSockets LAN with a z/VM Virtual Switch in IBM Documentation.
- See Scaling HyperPAV alias devices on Linux guests on z/VM for performance optimization.
- See Topics in LPAR performance for LPAR weight management and entitlements.
- Recommended host practices for IBM Z & IBM® LinuxONE environments

2.3.6. 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 HTTP or HTTPS server to establish a network connection to download their Ignition config files.

The machines are configured with static IP addresses. No DHCP server is required. Ensure that the machines have persistent IP addresses and hostnames.

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.

2.3.6.1. 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

Protocol	Port	Description
UDP	**4789**	VXLAN
	6081	Geneve
	9000-9999	Host level services, including the node exporter on ports **9100-9101**.
	500	IPsec IKE packets
	4500	IPsec NAT-T packets
	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 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* .

Additional resources

- Configuring chrony time service

2.3.7. 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.

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.

Component	Record	Description
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.

2.3.7.1. 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.

2.3.8. 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.

2.3.8.1. 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, preparing a web server for the Ignition files, 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. Set up static IP addresses.
2. Set up an HTTP or HTTPS server to provide Ignition files to the cluster nodes.
3. 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.
4. 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.

5. 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.

6. 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.

7. 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.

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

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.

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.

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.

2.7. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on your provisioning machine.

Prerequisites

- You have a machine that runs Linux, for example Red Hat Enterprise Linux 8, 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

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

Procedure

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

```
$ mkdir <installation_directory>
```

IMPORTANT

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

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

NOTE

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

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

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

Additional resources

- Installation configuration parameters for IBM Z

2.9.1. Sample install-config.yaml file for IBM Z

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
  architecture: s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture: s390x
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 available on your OpenShift Container Platform nodes, the **hyperthreading** parameter has no effect.

IMPORTANT

If you disable **hyperthreading**, whether on your OpenShift Container Platform nodes 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 IBM Z infrastructure.

> **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 Installing the system in 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.

2.9.2. Configuring the cluster-wide proxy during installation

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

Prerequisites

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

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

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

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

2.9.3. Configuring a three-node cluster

Optionally, you can deploy zero compute machines in a minimal three node 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.

> **NOTE**
>
> The preferred resource for control plane nodes is six vCPUs and 21 GB. For three control plane nodes this is the memory + vCPU equivalent of a minimum five-node cluster. You should back the three nodes, each installed on a 120 GB disk, with three IFLs that are SMT2 enabled. The minimum tested setup is three vCPUs and 10 GB on a 120 GB disk for each control plane node.

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. 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**.

2.10.1. Cluster Network Operator configuration object

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

Table 2.9. 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.

defaultNetwork object configuration

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

Table 2.10. defaultNetwork object

Field	Type	Description

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 2.11. **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**. This value cannot be changed after cluster installation.
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**.

Example OpenShift SDN configuration

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

Configuration for the OVN-Kubernetes network plugin

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

Table 2.12. **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 2.13. **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 2.14. **gatewayConfig** object

Field	Type	Description
routingViaHost	**boolean**	Set this field to **true** to send egress traffic from pods to the host networking stack. 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 2.15. **kubeProxyConfig** object

<table>
<tr><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td>iptablesSyncPeriod</td><td>string</td><td>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.</td></tr>
<tr><td>proxyArguments.iptables-min-sync-period</td><td>array</td><td>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:
<pre>kubeProxyConfig:
  proxyArguments:
    iptables-min-sync-period:
    - 0s</pre></td></tr>
</table>

2.11. 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.

NOTE

The installation program that generates the manifest and Ignition files is architecture specific and can be obtained from the client image mirror. The Linux version of the installation program runs on s390x only. This installer program is also available as a Mac OS version.

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
```

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

To install OpenShift Container Platform on IBM Z infrastructure that you provision, you must install Red Hat Enterprise Linux CoreOS (RHCOS) on z/VM guest virtual 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 z/VM guest virtual machines have rebooted.

Complete the following steps to create the machines.

Prerequisites

- An HTTP or HTTPS server running on your provisioning machine that is accessible to the machines you create.

Procedure

1. Log in to Linux on your provisioning machine.

2. Obtain the Red Hat Enterprise Linux CoreOS (RHCOS) kernel, initramfs, and rootfs files from the RHCOS image mirror.

   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. Only use the appropriate kernel, initramfs, and rootfs artifacts described in the following procedure.

   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**

> **NOTE**
>
> The rootfs image is the same for FCP and DASD.

3. Create parameter files. The following parameters are specific for a particular virtual machine:

- For **ip=**, specify the following seven entries:

  i. The IP address for the machine.

  ii. An empty string.

  iii. The gateway.

  iv. The netmask.

  v. The machine host and domain name in the form **hostname.domainname**. Omit this value to let RHCOS decide.

  vi. The network interface name. Omit this value to let RHCOS decide.

  vii. If you use static IP addresses, specify **none**.

- For **coreos.inst.ignition_url=**, specify the Ignition file for the machine role. Use **bootstrap.ign**, **master.ign**, or **worker.ign**. Only HTTP and HTTPS protocols are supported.

- For **coreos.live.rootfs_url=**, specify the matching rootfs artifact for the kernel and initramfs you are booting. Only HTTP and HTTPS protocols are supported.

- For installations on DASD-type disks, complete the following tasks:

  i. For **coreos.inst.install_dev=**, specify **dasda**.

  ii. Use **rd.dasd=** to specify the DASD where RHCOS is to be installed.

  iii. Leave all other parameters unchanged.
  Example parameter file, **bootstrap-0.parm**, for the bootstrap machine:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=dasda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-
rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/bootstrap.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

  Write all options in the parameter file as a single line and make sure you have no newline characters.

- For installations on FCP-type disks, complete the following tasks:

  i. Use **rd.zfcp=<adapter>,<wwpn>,<lun>** to specify the FCP disk where RHCOS is to be installed. For multipathing repeat this step for each additional path.

> **NOTE**
>
> When you install with multiple paths, you must enable multipathing directly after the installation, not at a later point in time, as this can cause problems.

ii. Set the install device as: **coreos.inst.install_dev=sda**.

> **NOTE**
>
> If additional LUNs are configured with NPIV, FCP requires **zfcp.allow_lun_scan=0**. If you must enable **zfcp.allow_lun_scan=1** because you use a CSI driver, for example, you must configure your NPIV so that each node cannot access the boot partition of another node.

iii. Leave all other parameters unchanged.

> **IMPORTANT**
>
> Additional postinstallation steps are required to fully enable multipathing. For more information, see “Enabling multipathing with kernel arguments on RHCOS" in *Post-installation machine configuration tasks*.

The following is an example parameter file **worker-1.parm** for a worker node with multipathing:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=sda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

Write all options in the parameter file as a single line and make sure you have no newline characters.

4. Transfer the initramfs, kernel, parameter files, and RHCOS images to z/VM, for example with FTP. For details about how to transfer the files with FTP and boot from the virtual reader, see Installing under Z/VM.

5. Punch the files to the virtual reader of the z/VM guest virtual machine that is to become your bootstrap node.
See PUNCH in IBM Documentation.

TIP

You can use the CP PUNCH command or, if you use Linux, the **vmur** command to transfer files between two z/VM guest virtual machines.

6. Log in to CMS on the bootstrap machine.

7. IPL the bootstrap machine from the reader:

```
$ ipl c
```

See IPL in IBM Documentation.

8. Repeat this procedure for the other machines in the cluster.

2.12.1. 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.

2.12.1.1. 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,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

Always set option **fail_over_mac=1** in active-backup mode, to avoid problems when shared OSA/RoCE cards are used.

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
```

2.13. 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.

2.14. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

```
$ export KUBECONFIG=&lt;installation_directory&gt;/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
```

2.15. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

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

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.25.0
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-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

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.

2.16. INITIAL OPERATOR CONFIGURATION

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

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    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.

2.16.1. Image registry storage configuration

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

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

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

2.16.1.1. Configuring registry storage for IBM Z

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 on IBM Z.
- 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
     ```

2.16.1.2. 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.

2.17. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE

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

Prerequisites

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

Procedure

1. Confirm that all the cluster components are online 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.

2.18. 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
- How to generate SOSREPORT within OpenShift4 nodes without SSH .

2.19. NEXT STEPS

- Enabling multipathing with kernel arguments on RHCOS .
- Customize your cluster.
- If necessary, you can opt out of remote health reporting .

CHAPTER 3. INSTALLING A CLUSTER WITH Z/VM ON IBM Z AND IBM(R) LINUXONE IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.12, you can install a cluster on IBM Z or IBM® LinuxONE infrastructure that you provision in a restricted network.

> **NOTE**
>
> While this document refers to only IBM Z, all information in it also applies to IBM® LinuxONE.

> **IMPORTANT**
>
> Additional considerations exist for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you install an OpenShift Container Platform 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.
- You created a mirror registry for installation in a restricted network and obtained the **imageContentSources** data for your version of OpenShift Container Platform.
- Before you begin the installation process, you must move or remove any existing installation files. This ensures that the required installation files are created and updated during the installation process.

  > **IMPORTANT**
  >
  > Ensure that installation steps are done from a machine with access to the installation media.

- You provisioned persistent storage using OpenShift Data Foundation or other supported storage protocols for your cluster. To deploy a private image registry, you must set up persistent storage with **ReadWriteMany** access.
- 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.

3.2. 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.

3.2.1. Additional limits

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

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

3.3. 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.

3.4. 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.

3.4.1. 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.

IMPORTANT

To improve high availability of your cluster, distribute the control plane machines over different z/VM instances on at least two physical machines.

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

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

3.4.2. Minimum resource requirements for cluster installation

Each cluster machine must meet the following minimum requirements:

Table 3.2. Minimum resource requirements

Machine	Operating System	vCPU [1]	Virtual RAM	Storage	IOPS
Bootstrap	RHCOS	4	16 GB	100 GB	N/A
Control plane	RHCOS	4	16 GB	100 GB	N/A

Machine	Operating System	vCPU [1]	Virtual RAM	Storage	IOPS
Compute	RHCOS	2	8 GB	100 GB	N/A

1. One physical core (IFL) provides two logical cores (threads) when SMT-2 is enabled. The hypervisor can provide two or more vCPUs.

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

3.4.3. Minimum IBM Z system environment

You can install OpenShift Container Platform version 4.12 on the following IBM hardware:

- IBM z16 (all models), IBM z15 (all models), IBM z14 (all models), IBM z13, and IBM z13s
- IBM® LinuxONE Emperor 4, IBM® LinuxONE III (all models), IBM® LinuxONE Emperor II, IBM® LinuxONE Rockhopper II, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper

> **NOTE**
>
> Support for RHCOS functionality for IBM z13 all models, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper is deprecated. These hardware models remain fully supported in OpenShift Container Platform 4.12. However, Red Hat recommends that you use later hardware models.

Hardware requirements

- The equivalent of six Integrated Facilities for Linux (IFL), which are SMT2 enabled, for each cluster.
- At least one network connection to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.

> **NOTE**
>
> You can use dedicated or shared IFLs to assign sufficient compute resources. Resource sharing is one of the key strengths of IBM Z. However, you must adjust capacity correctly on each hypervisor layer and ensure sufficient resources for every OpenShift Container Platform cluster.

> **IMPORTANT**
>
> Since the overall performance of the cluster can be impacted, the LPARs that are used to set up the OpenShift Container Platform clusters must provide sufficient compute capacity. In this context, LPAR weight management, entitlements, and CPU shares on the hypervisor level play an important role.

Operating system requirements

- One instance of z/VM 7.2 or later

On your z/VM instance, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines
- Two guest virtual machines for OpenShift Container Platform compute machines
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

IBM Z network connectivity requirements

To install on IBM Z under z/VM, you require a single z/VM virtual NIC in layer 2 mode. You also need:

- A direct-attached OSA or RoCE network adapter
- A z/VM VSwitch set up. For a preferred setup, use OSA link aggregation.

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

3.4.4. Preferred IBM Z system environment

Hardware requirements

- Three LPARS that each have the equivalent of six IFLs, which are SMT2 enabled, for each cluster.
- Two network connections to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.
- HiperSockets, which are attached to a node either directly as a device or by bridging with one z/VM VSWITCH to be transparent to the z/VM guest. To directly connect HiperSockets to a node, you must set up a gateway to the external network via a RHEL 8 guest to bridge to the HiperSockets network.

Operating system requirements

- Two or three instances of z/VM 7.2 or later for high availability

On your z/VM instances, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines, one per z/VM instance.
- At least six guest virtual machines for OpenShift Container Platform compute machines, distributed across the z/VM instances.
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine.
- To ensure the availability of integral components in an overcommitted environment, increase the priority of the control plane by using the CP command **SET SHARE**. Do the same for infrastructure nodes, if they exist. See SET SHARE in IBM Documentation.

IBM Z network connectivity requirements

To install on IBM Z under z/VM, you require a single z/VM virtual NIC in layer 2 mode. You also need:

- A direct-attached OSA or RoCE network adapter
- A z/VM VSwitch set up. For a preferred setup, use OSA link aggregation.

Disk storage for the z/VM guest virtual machines

- FICON attached disk storage (DASDs). These can be z/VM minidisks, fullpack minidisks, or dedicated DASDs, all of which must be formatted as CDL, which is the default. To reach the minimum required DASD size for Red Hat Enterprise Linux CoreOS (RHCOS) installations, you need extended address volumes (EAV). If available, use HyperPAV and High Performance FICON (zHPF) to ensure optimal performance.
- FCP attached disk storage

Storage / Main Memory

- 16 GB for OpenShift Container Platform control plane machines
- 8 GB for OpenShift Container Platform compute machines
- 16 GB for the temporary OpenShift Container Platform bootstrap machine

3.4.5. 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 Bridging a HiperSockets LAN with a z/VM Virtual Switch in IBM Documentation.
- See Scaling HyperPAV alias devices on Linux guests on z/VM for performance optimization.
- See Topics in LPAR performance for LPAR weight management and entitlements.
- Recommended host practices for IBM Z & IBM® LinuxONE environments

3.4.6. 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.

3.4.6.1. 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.

3.4.6.2. 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 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* .

Additional resources

- Configuring chrony time service

3.4.7. 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.

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.

3.4.7.1. 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

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.

3.4.8. 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
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

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

3.4.8.1. 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.

3.5. 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, preparing a web server for the Ignition files, 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. Set up static IP addresses.
2. Set up an HTTP or HTTPS server to provide Ignition files to the cluster nodes.
3. 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.
4. 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.

5. 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.
6. 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.

7. 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.

3.6. 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.

3.7. 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.

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.

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

Procedure

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

```
$ mkdir <installation_directory>
```

IMPORTANT

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

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

NOTE

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

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

IMPORTANT

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

Additional resources

- Installation configuration parameters for IBM Z

3.8.1. Sample install-config.yaml file for IBM Z

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
  architecture: s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture: s390x
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-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 18
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  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 available on your OpenShift Container Platform nodes, the **hyperthreading** parameter has no effect.

> **IMPORTANT**
>
> If you disable **hyperthreading**, whether on your OpenShift Container Platform nodes 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**,

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 IBM Z infrastructure.

> **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 Installing the system in 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 Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry. The certificate file can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

18 Provide the **imageContentSources** section from the output of the command to mirror the repository.

3.8.2. Configuring the cluster-wide proxy during installation

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

Prerequisites

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

> **NOTE**
>
> The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.
>
> For installations on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

Procedure

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

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

3.8.3. Configuring a three-node cluster

Optionally, you can deploy zero compute machines in a minimal three node 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.

NOTE

The preferred resource for control plane nodes is six vCPUs and 21 GB. For three control plane nodes this is the memory + vCPU equivalent of a minimum five-node cluster. You should back the three nodes, each installed on a 120 GB disk, with three IFLs that are SMT2 enabled. The minimum tested setup is three vCPUs and 10 GB on a 120 GB disk for each control plane node.

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.

3.9. 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**.

3.9.1. Cluster Network Operator configuration object

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

Table 3.9. Cluster Network Operator configuration object

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

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.

defaultNetwork object configuration

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

Table 3.10. **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 3.11. **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.
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**. This value cannot be changed after cluster installation.

Field	Type	Description
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**.

Example OpenShift SDN configuration

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

Configuration for the OVN-Kubernetes network plugin

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

Table 3.12. **ovnKubernetesConfig** object

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.

Field	Type	Description
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 3.13. **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 3.14. **gatewayConfig** object

Field	Type	Description
routingViaHost	**boolean**	Set this field to **true** to send egress traffic from pods to the host networking stack. 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 3.15. **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`

3.10. 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.

NOTE

The installation program that generates the manifest and Ignition files is architecture specific and can be obtained from the client image mirror. The Linux version of the installation program runs on s390x only. This installer program is also available as a Mac OS version.

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
```

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

To install OpenShift Container Platform on IBM Z infrastructure that you provision, you must install Red Hat Enterprise Linux CoreOS (RHCOS) on z/VM guest virtual 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 z/VM guest virtual machines have rebooted.

Complete the following steps to create the machines.

Prerequisites

- An HTTP or HTTPS server running on your provisioning machine that is accessible to the machines you create.

Procedure

1. Log in to Linux on your provisioning machine.

2. Obtain the Red Hat Enterprise Linux CoreOS (RHCOS) kernel, initramfs, and rootfs files from the RHCOS image mirror.

> **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. Only use the appropriate kernel, initramfs, and rootfs artifacts described in the following procedure.

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**

> NOTE
>
> The rootfs image is the same for FCP and DASD.

3. Create parameter files. The following parameters are specific for a particular virtual machine:
   - For **ip=**, specify the following seven entries:
     i. The IP address for the machine.
     ii. An empty string.
     iii. The gateway.
     iv. The netmask.
     v. The machine host and domain name in the form **hostname.domainname**. Omit this value to let RHCOS decide.
     vi. The network interface name. Omit this value to let RHCOS decide.
     vii. If you use static IP addresses, specify **none**.
   - For **coreos.inst.ignition_url=**, specify the Ignition file for the machine role. Use **bootstrap.ign**, **master.ign**, or **worker.ign**. Only HTTP and HTTPS protocols are supported.
   - For **coreos.live.rootfs_url=**, specify the matching rootfs artifact for the kernel and initramfs you are booting. Only HTTP and HTTPS protocols are supported.
   - For installations on DASD-type disks, complete the following tasks:
     i. For **coreos.inst.install_dev=**, specify **dasda**.
     ii. Use **rd.dasd=** to specify the DASD where RHCOS is to be installed.
     iii. Leave all other parameters unchanged.
     Example parameter file, **bootstrap-0.parm**, for the bootstrap machine:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=dasda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-
rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/bootstrap.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.dasd=0.0.3490
```

Write all options in the parameter file as a single line and make sure you have no newline characters.

- For installations on FCP-type disks, complete the following tasks:

  i. Use **rd.zfcp=<adapter>,<wwpn>,<lun>** to specify the FCP disk where RHCOS is to be installed. For multipathing repeat this step for each additional path.

  NOTE

  When you install with multiple paths, you must enable multipathing directly after the installation, not at a later point in time, as this can cause problems.

  ii. Set the install device as: **coreos.inst.install_dev=sda**.

  NOTE

  If additional LUNs are configured with NPIV, FCP requires **zfcp.allow_lun_scan=0**. If you must enable **zfcp.allow_lun_scan=1** because you use a CSI driver, for example, you must configure your NPIV so that each node cannot access the boot partition of another node.

  iii. Leave all other parameters unchanged.

  IMPORTANT

  Additional postinstallation steps are required to fully enable multipathing. For more information, see “Enabling multipathing with kernel arguments on RHCOS" in *Post-installation machine configuration tasks*.

The following is an example parameter file **worker-1.parm** for a worker node with multipathing:

```
rd.neednet=1 \
console=ttysclp0 \
coreos.inst.install_dev=sda \
coreos.live.rootfs_url=http://cl1.provide.example.com:8080/assets/rhcos-live-
rootfs.s390x.img \
coreos.inst.ignition_url=http://cl1.provide.example.com:8080/ignition/worker.ign \
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 \
zfcp.allow_lun_scan=0 \
rd.zfcp=0.0.1987,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763070bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.1987,0x50050763071bc5e3,0x4008400B00000000 \
rd.zfcp=0.0.19C7,0x50050763071bc5e3,0x4008400B00000000
```

Write all options in the parameter file as a single line and make sure you have no newline characters.

4. Transfer the initramfs, kernel, parameter files, and RHCOS images to z/VM, for example with FTP. For details about how to transfer the files with FTP and boot from the virtual reader, see Installing under Z/VM.

5. Punch the files to the virtual reader of the z/VM guest virtual machine that is to become your bootstrap node.
   See PUNCH in IBM Documentation.

   TIP

   You can use the CP PUNCH command or, if you use Linux, the **vmur** command to transfer files between two z/VM guest virtual machines.

6. Log in to CMS on the bootstrap machine.

7. IPL the bootstrap machine from the reader:

   ```
   $ ipl c
   ```

   See IPL in IBM Documentation.

8. Repeat this procedure for the other machines in the cluster.

3.11.1. 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.

3.11.1.1. 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,fail_over_mac=1
ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none
```

Always set option **fail_over_mac=1** in active-backup mode, to avoid problems when shared OSA/RoCE cards are used.

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
```

3.12. 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.

3.13. 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
```

3.14. 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.

3.15. 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.

3.15.1. 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.

3.15.2. 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.

3.15.2.1. Configuring registry storage for IBM Z

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 on IBM Z.
- 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
    ```

3.15.2.2. 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.

3.16. 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.

Additional resources

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH.

3.17. NEXT STEPS

- Customize your cluster.
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.
- If necessary, you can opt out of remote health reporting .
- If necessary, see Registering your disconnected cluster

CHAPTER 4. INSTALLING A CLUSTER WITH RHEL KVM ON IBM Z AND IBM(R) LINUXONE

In OpenShift Container Platform version 4.12, you can install a cluster on IBM Z or IBM® LinuxONE infrastructure that you provision.

> **NOTE**
>
> While this document refers only to IBM Z, all information in it also applies to IBM® LinuxONE.

> **IMPORTANT**
>
> Additional considerations exist for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you install an OpenShift Container Platform cluster.

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.
- Before you begin the installation process, you must clean the installation directory. This ensures that the required installation files are created and updated during the installation process.
- You provisioned persistent storage using OpenShift Data Foundation or other supported storage protocols for your cluster. To deploy a private image registry, you must set up persistent storage with **ReadWriteMany** access.
- 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.

- You provisioned a RHEL Kernel Virtual Machine (KVM) system that is hosted on the logical partition (LPAR) and based on RHEL 8.4 or later. See Red Hat Enterprise Linux 8 and 9 Life Cycle.

4.2. 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.

4.3. MACHINE REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

One or more KVM host machines based on RHEL 8.4 or later. Each RHEL KVM host machine must have libvirt installed and running. The virtual machines are provisioned under each RHEL KVM host machine.

4.3.1. Required machines

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.

IMPORTANT

To improve high availability of your cluster, distribute the control plane machines over different RHEL instances on at least two physical machines.

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

See Red Hat Enterprise Linux technology capabilities and limits .

4.3.2. Network connectivity requirements

The OpenShift Container Platform installer creates the Ignition files, which are necessary for all the Red Hat Enterprise Linux CoreOS (RHCOS) virtual machines. The automated installation of OpenShift Container Platform is performed by the bootstrap machine. It starts the installation of OpenShift Container Platform on each node, starts the Kubernetes cluster, and then finishes. During this bootstrap, the virtual machine must have an established network connection either through a Dynamic Host Configuration Protocol (DHCP) server or static IP address.

4.3.3. IBM Z network connectivity requirements

To install on IBM Z under RHEL KVM, you need:

- A RHEL KVM host configured with an OSA or RoCE network adapter.
- Either a RHEL KVM host that is configured to use bridged networking in libvirt or MacVTap to connect the network to the guests.
  See Types of virtual network connections.

4.3.4. Host machine resource requirements

The RHEL KVM host in your environment must meet the following requirements to host the virtual machines that you plan for the OpenShift Container Platform environment. See Getting started with virtualization.

You can install OpenShift Container Platform version 4.12 on the following IBM hardware:

- IBM z16 (all models), IBM z15 (all models), IBM z14 (all models), IBM z13, and IBM z13s
- IBM® LinuxONE Emperor 4, IBM® LinuxONE III (all models), IBM® LinuxONE Emperor II, IBM® LinuxONE Rockhopper II, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper

NOTE

Support for RHCOS functionality for IBM z13 all models, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper is deprecated. These hardware models remain fully supported in OpenShift Container Platform 4.12. However, Red Hat recommends that you use later hardware models.

4.3.5. Minimum IBM Z system environment

Hardware requirements

- The equivalent of six Integrated Facilities for Linux (IFL), which are SMT2 enabled, for each cluster.
- At least one network connection to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.

NOTE

You can use dedicated or shared IFLs to assign sufficient compute resources. Resource sharing is one of the key strengths of IBM Z. However, you must adjust capacity correctly on each hypervisor layer and ensure sufficient resources for every OpenShift Container Platform cluster.

IMPORTANT

Since the overall performance of the cluster can be impacted, the LPARs that are used to set up the OpenShift Container Platform clusters must provide sufficient compute capacity. In this context, LPAR weight management, entitlements, and CPU shares on the hypervisor level play an important role.

Operating system requirements

- One LPAR running on RHEL 8.4 or later with KVM, which is managed by libvirt

On your RHEL KVM host, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines
- Two guest virtual machines for OpenShift Container Platform compute machines
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

4.3.6. Minimum resource requirements

Each cluster virtual machine must meet the following minimum requirements:

Virtual Machine	Operating System	vCPU [1]	Virtual RAM	Storage	IOPS
Bootstrap	RHCOS	4	16 GB	100 GB	N/A
Control plane	RHCOS	4	16 GB	100 GB	N/A
Compute	RHCOS	2	8 GB	100 GB	N/A

1. One physical core (IFL) provides two logical cores (threads) when SMT-2 is enabled. The hypervisor can provide two or more vCPUs.

4.3.7. Preferred IBM Z system environment

Hardware requirements

- Three LPARS that each have the equivalent of six IFLs, which are SMT2 enabled, for each cluster.
- Two network connections to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.

Operating system requirements

- For high availability, two or three LPARs running on RHEL 8.4 or later with KVM, which are managed by libvirt.

On your RHEL KVM host, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines, distributed across the RHEL KVM host machines.

- At least six guest virtual machines for OpenShift Container Platform compute machines, distributed across the RHEL KVM host machines.
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine.
- To ensure the availability of integral components in an overcommitted environment, increase the priority of the control plane by using **cpu_shares**. Do the same for infrastructure nodes, if they exist. See schedinfo in IBM Documentation.

4.3.8. Preferred resource requirements

The preferred requirements for each cluster virtual machine are:

Virtual Machine	Operating System	vCPU	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	8	16 GB	120 GB
Compute	RHCOS	6	8 GB	120 GB

4.3.9. 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

- Recommended host practices for IBM Z & IBM® LinuxONE environments

4.3.10. 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.

4.3.10.1. 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.

4.3.10.2. 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.

NOTE

The RHEL KVM host must be configured to use bridged networking in libvirt or MacVTap to connect the network to the virtual machines. The virtual machines must have access to the network, which is attached to the RHEL KVM host. Virtual Networks, for example network address translation (NAT), within KVM are not a supported configuration.

Table 4.2. 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 4.3. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 4.4. 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

4.3.11. 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.

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 4.5. 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.

4.3.11.1. 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.

4.3.12. 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 4.6. 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

Port	Back-end machines (pool members)	Internal	External	Description
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 4.7. Application Ingress load balancer

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

4.3.12.1. 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.

4.4. 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. Choose to perform either a fast track installation of Red Hat Enterprise Linux CoreOS (RHCOS) or a full installation of Red Hat Enterprise Linux CoreOS (RHCOS). For the full installation, you must set up an HTTP or HTTPS server to provide Ignition files and install images to the cluster nodes. For the fast track installation an HTTP or HTTPS server is not required, however, a DHCP

server is required. See sections “Fast-track installation: Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines" and “Full installation: Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines".

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

4.5. 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.

4.6. 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.

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.

4.7. OBTAINING THE INSTALLATION PROGRAM

Before you install OpenShift Container Platform, download the installation file on your provisioning machine.

Prerequisites

- You have a machine that runs Linux, for example Red Hat Enterprise Linux 8, 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.

4.8. 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>
```

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

Procedure

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

```
$ mkdir <installation_directory>
```

IMPORTANT

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

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

NOTE

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

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

IMPORTANT

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

Additional resources

- Installation configuration parameters for IBM Z

4.9.1. Sample install-config.yaml file for IBM Z

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
  architecture: s390x
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
  architecture: s390x
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 available on your OpenShift Container Platform nodes, the **hyperthreading** parameter has no effect.

> **IMPORTANT**
>
> If you disable **hyperthreading**, whether on your OpenShift Container Platform nodes 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 IBM Z infrastructure.

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 Installing the system in 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.

4.9.2. Configuring the cluster-wide proxy during installation

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

Prerequisites

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

NOTE

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

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

Procedure

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

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

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.

4.9.3. Configuring a three-node cluster

Optionally, you can deploy zero compute machines in a minimal three node 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.

> **NOTE**
>
> The preferred resource for control plane nodes is six vCPUs and 21 GB. For three control plane nodes this is the memory + vCPU equivalent of a minimum five-node cluster. You should back the three nodes, each installed on a 120 GB disk, with three IFLs that are SMT2 enabled. The minimum tested setup is three vCPUs and 10 GB on a 120 GB disk for each control plane node.

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.

4.10. CLUSTER NETWORK OPERATOR CONFIGURATION

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a 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**.

4.10.1. Cluster Network Operator configuration object

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

Table 4.8. Cluster Network Operator configuration object

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

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.

defaultNetwork object configuration

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

Table 4.9. **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 4.10. **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**. This value cannot be changed after cluster installation.
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**.

Example OpenShift SDN configuration

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

Configuration for the OVN-Kubernetes network plugin

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

Table 4.11. 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 4.12. **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 4.13. **gatewayConfig** object

Field	Type	Description
routingViaHost	**boolean**	Set this field to **true** to send egress traffic from pods to the host networking stack. 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 4.14. **kubeProxyConfig** object

<table>
<tr><th>Field</th><th>Type</th><th>Description</th></tr>
<tr><td>iptablesSyncPeriod</td><td>string</td><td>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.</td></tr>
<tr><td>proxyArguments.iptables-min-sync-period</td><td>array</td><td>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:
<pre>kubeProxyConfig:
  proxyArguments:
    iptables-min-sync-period:
    - 0s</pre></td></tr>
</table>

4.11. 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.

> **NOTE**
>
> The installation program that generates the manifest and Ignition files is architecture specific and can be obtained from the client image mirror. The Linux version of the installation program runs on s390x only. This installer program is also available as a Mac OS version.

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
```

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

To install OpenShift Container Platform on IBM Z infrastructure that you provision, you must install Red Hat Enterprise Linux CoreOS (RHCOS) as Red Hat Enterprise Linux (RHEL) guest virtual 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.

You can perform a fast-track installation of RHCOS that uses a prepackaged QEMU copy-on-write (QCOW2) disk image. Alternatively, you can perform a full installation on a new QCOW2 disk image.

To add further security to your system, you can optionally install RHCOS using IBM Secure Execution before proceeding to the fast-track installation.

4.12.1. Installing RHCOS using IBM Secure Execution

Before you install RHCOS using IBM Secure Execution, you must prepare the underlying infrastructure.

IMPORTANT

Installing RHCOS using IBM Secure Execution is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

Prerequisites

- IBM z15 or later, or IBM® LinuxONE III or later.
- Red Hat Enterprise Linux (RHEL) 8 or later.

- You have a bootstrap Ignition file. The file is not protected, enabling others to view and edit it.
- You have verified that the boot image has not been altered after installation.
- You must run all your nodes as IBM Secure Execution guests.

Procedure

1. Prepare your RHEL KVM host to support IBM Secure Execution.
   - By default, KVM hosts do not support guests in IBM Secure Execution mode. To support guests in IBM Secure Execution mode, KVM hosts must boot in LPAR mode with the kernel parameter specification **prot_virt=1**. To enable **prot_virt=1** on RHEL 8, follow these steps:
     a. Navigate to **/boot/loader/entries/** to modify your bootloader configuration file ***.conf**.
     b. Add the kernel command line parameter **prot_virt=1**.
     c. Run the **zipl** command and reboot your system.
        KVM hosts that successfully start with support for IBM Secure Execution for Linux issue the following kernel message:

        ```
        prot_virt: Reserving <amount>MB as ultravisor base storage.
        ```

     d. To verify that the KVM host now supports IBM Secure Execution, run the following command:

        ```
        # cat /sys/firmware/uv/prot_virt_host
        ```

        #### Example output

        ```
        1
        ```

        The value of this attribute is 1 for Linux instances that detect their environment as consistent with that of a secure host. For other instances, the value is 0.

2. Add your host keys to the KVM guest via Ignition.
   During the first boot, RHCOS looks for your host keys to re-encrypt itself with them. RHCOS searches for files starting with **ibm-z-hostkey-** in the **/etc/se-hostkeys** directory. All host keys, for each machine the cluster is running on, must be loaded into the directory by the administrator. After first boot, you cannot run the VM on any other machines.

   > **NOTE**
   >
   > You need to prepare your Ignition file on a safe system. For example, another IBM Secure Execution guest.

   For example:

   ```
   {
     "ignition": { "version": "3.0.0" },
     "storage": {
       "files": [
         {
   ```

````
        "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
        "contents": {
          "source": "data:;base64,<base64 encoded hostkey document>"
        },
        "mode": 420
      },
      {
        "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
        "contents": {
          "source": "data:;base64,<base64 encoded hostkey document>"
        },
        "mode": 420
      }
    ]
  }
}
```
````

NOTE

You can add as many host keys as required if you want your node to be able to run on multiple IBM Z machines.

3. To generate the Base64 encoded string, run the following command:

```
base64 <your-hostkey>.crt
```

Compared to guests not running IBM Secure Execution, the first boot of the machine is longer because the entire image is encrypted with a randomly generated LUKS passphrase before the Ignition phase.

4. Follow the fast-track installation procedure to install nodes using the IBM Secure Exection QCOW image.

Additional resources

- Introducing IBM Secure Execution for Linux
- Linux as an IBM Secure Execution host or guest

4.12.2. Fast-track installation by using a prepackaged QCOW2 disk image

Complete the following steps to create the machines in a fast-track installation of Red Hat Enterprise Linux CoreOS (RHCOS), importing a prepackaged Red Hat Enterprise Linux CoreOS (RHCOS) QEMU copy-on-write (QCOW2) disk image.

Prerequisites

- At least one LPAR running on RHEL 8.4 or later with KVM, referred to as RHEL KVM host in this procedure.
- The KVM/QEMU hypervisor is installed on the RHEL KVM host.
- A domain name server (DNS) that can perform hostname and reverse lookup for the nodes.
```

- A DHCP server that provides IP addresses.

Procedure

1. Obtain the RHEL QEMU copy-on-write (QCOW2) disk image file from the Product Downloads page on the Red Hat Customer Portal or from the RHCOS image mirror page.

> **IMPORTANT**
>
> The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Only use the appropriate RHCOS QCOW2 image described in the following procedure.

2. Download the QCOW2 disk image and Ignition files to a common directory on the RHEL KVM host.
For example: **/var/lib/libvirt/images**

> **NOTE**
>
> The Ignition files are generated by the OpenShift Container Platform installer.

3. Create a new disk image with the QCOW2 disk image backing file for each KVM guest node.

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Create the new KVM guest nodes using the Ignition file and the new disk image.

```
$ virt-install --noautoconsole \
  --connect qemu:///system \
  --name {vn_name} \
  --memory {memory} \
  --vcpus {vcpus} \
  --disk {disk} \
  --import \
  --network network={network},mac={mac} \
  --disk path={ign_file},format=raw,readonly=on,serial=ignition,startup_policy=optional
```

4.12.3. Full installation on a new QCOW2 disk image

Complete the following steps to create the machines in a full installation on a new QEMU copy-on-write (QCOW2) disk image.

Prerequisites

- At least one LPAR running on RHEL 8.4 or later with KVM, referred to as RHEL KVM host in this procedure.
- The KVM/QEMU hypervisor is installed on the RHEL KVM host.
- A domain name server (DNS) that can perform hostname and reverse lookup for the nodes.

- An HTTP or HTTPS server is set up.

Procedure

1. Obtain the RHEL kernel, initramfs, and rootfs files from the Product Downloads page on the Red Hat Customer Portal or from the RHCOS image mirror page.

   IMPORTANT

   The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Only use the appropriate RHCOS QCOW2 image described in the following procedure.

   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**

2. Move the downloaded RHEL live kernel, initramfs, and rootfs as well as the Ignition files to an HTTP or HTTPS server before you launch **virt-install**.

   NOTE

   The Ignition files are generated by the OpenShift Container Platform installer.

3. Create the new KVM guest nodes using the RHEL kernel, initramfs, and Ignition files, the new disk image, and adjusted parm line arguments.

   - For **--location**, specify the location of the kernel/initrd on the HTTP or HTTPS server.
   - For **coreos.inst.ignition_url=**, specify the Ignition file for the machine role. Use **bootstrap.ign**, **master.ign**, or **worker.ign**. Only HTTP and HTTPS protocols are supported.
   - For **coreos.live.rootfs_url=**, specify the matching rootfs artifact for the kernel and initramfs you are booting. Only HTTP and HTTPS protocols are supported.

```
$ virt-install \
   --connect qemu:///system \
   --name {vn_name} \
   --vcpus {vcpus} \
   --memory {memory_mb} \
   --disk {vn_name}.qcow2,size={image_size| default(10,true)} \
   --network network={virt_network_parm} \
   --boot hd \
   --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
   --extra-args "rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=vda
coreos.live.rootfs_url={rhcos_liveos} ip={ip}::{default_gateway}:{subnet_mask_length}:
{vn_name}:enc1:none:{MTU} nameserver={dns} coreos.inst.ignition_url={rhcos_ign}" \
   --noautoconsole \
   --wait
```

4.12.4. Advanced RHCOS installation reference

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

4.12.4.1. Networking 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 on your RHCOS nodes for ISO installations. The examples describe how to use the **ip=** and **nameserver=** kernel arguments.

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

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
```

4.13. 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.

4.14. LOGGING IN TO THE CLUSTER BY USING THE CLI

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

Prerequisites

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

Procedure

1. Export the **kubeadmin** credentials:

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

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

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

```
$ oc whoami
```

Example output

```
system:admin
```

4.15. APPROVING THE CERTIFICATE SIGNING REQUESTS FOR YOUR MACHINES

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

Prerequisites

- You added machines to your cluster.

Procedure

1. Confirm that the cluster recognizes the machines:

```
$ oc get nodes
```

Example output

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.25.0
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-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

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.

4.16. INITIAL OPERATOR CONFIGURATION

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

Prerequisites

- Your control plane has initialized.

Procedure

1. Watch the cluster components come online:

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

Example output

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.12.0    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.

4.16.1. Image registry storage configuration

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

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

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

4.16.1.1. Configuring registry storage for IBM Z

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 on IBM Z.
- 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
```

4.16.1.2. 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.

4.17. COMPLETING INSTALLATION ON USER-PROVISIONED INFRASTRUCTURE

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

Prerequisites

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

Procedure

1. Confirm that all the cluster components are online 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.

4.18. 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
- How to generate SOSREPORT within OpenShift4 nodes without SSH .

4.19. NEXT STEPS

- Customize your cluster.
- If necessary, you can opt out of remote health reporting .

CHAPTER 5. INSTALLING A CLUSTER WITH RHEL KVM ON IBM Z AND IBM(R) LINUXONE IN A RESTRICTED NETWORK

In OpenShift Container Platform version 4.12, you can install a cluster on IBM Z or IBM® LinuxONE infrastructure that you provision in a restricted network.

> **NOTE**
>
> While this document refers to only IBM Z, all information in it also applies to IBM® LinuxONE.

> **IMPORTANT**
>
> Additional considerations exist for non-bare metal platforms. Review the information in the guidelines for deploying OpenShift Container Platform on non-tested platforms before you install an OpenShift Container Platform cluster.

5.1. 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.
- You must move or remove any existing installation files, before you begin the installation process. This ensures that the required installation files are created and updated during the installation process.

  > **IMPORTANT**
  >
  > Ensure that installation steps are done from a machine with access to the installation media.

- You provisioned persistent storage using OpenShift Data Foundation or other supported storage protocols for your cluster. To deploy a private image registry, you must set up persistent storage with **ReadWriteMany** access.
- 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.

- You provisioned a RHEL Kernel Virtual Machine (KVM) system that is hosted on the logical partition (LPAR) and based on RHEL 8.4 or later. See Red Hat Enterprise Linux 8 and 9 Life Cycle.

5.2. 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.

5.2.1. 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.

5.3. 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.

5.4. MACHINE REQUIREMENTS FOR A CLUSTER WITH USER-PROVISIONED INFRASTRUCTURE

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

One or more KVM host machines based on RHEL 8.4 or later. Each RHEL KVM host machine must have libvirt installed and running. The virtual machines are provisioned under each RHEL KVM host machine.

5.4.1. Required machines

The smallest OpenShift Container Platform clusters require the following hosts:

Table 5.1. Minimum required hosts

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

IMPORTANT

To improve high availability of your cluster, distribute the control plane machines over different RHEL instances on at least two physical machines.

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

See Red Hat Enterprise Linux technology capabilities and limits .

5.4.2. Network connectivity requirements

The OpenShift Container Platform installer creates the Ignition files, which are necessary for all the Red Hat Enterprise Linux CoreOS (RHCOS) virtual machines. The automated installation of OpenShift Container Platform is performed by the bootstrap machine. It starts the installation of OpenShift Container Platform on each node, starts the Kubernetes cluster, and then finishes. During this bootstrap, the virtual machine must have an established network connection either through a Dynamic Host Configuration Protocol (DHCP) server or static IP address.

5.4.3. IBM Z network connectivity requirements

To install on IBM Z under RHEL KVM, you need:

- A RHEL KVM host configured with an OSA or RoCE network adapter.

- Either a RHEL KVM host that is configured to use bridged networking in libvirt or MacVTap to connect the network to the guests.
  See Types of virtual network connections .

5.4.4. Host machine resource requirements

The RHEL KVM host in your environment must meet the following requirements to host the virtual machines that you plan for the OpenShift Container Platform environment. See Getting started with virtualization.

You can install OpenShift Container Platform version 4.12 on the following IBM hardware:

- IBM z16 (all models), IBM z15 (all models), IBM z14 (all models), IBM z13, and IBM z13s
- IBM® LinuxONE Emperor 4, IBM® LinuxONE III (all models), IBM® LinuxONE Emperor II, IBM® LinuxONE Rockhopper II, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper

NOTE

Support for RHCOS functionality for IBM z13 all models, IBM® LinuxONE Emperor, and IBM® LinuxONE Rockhopper is deprecated. These hardware models remain fully supported in OpenShift Container Platform 4.12. However, Red Hat recommends that you use later hardware models.

5.4.5. Minimum IBM Z system environment

Hardware requirements

- The equivalent of six Integrated Facilities for Linux (IFL), which are SMT2 enabled, for each cluster.
- At least one network connection to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.

NOTE

You can use dedicated or shared IFLs to assign sufficient compute resources. Resource sharing is one of the key strengths of IBM Z. However, you must adjust capacity correctly on each hypervisor layer and ensure sufficient resources for every OpenShift Container Platform cluster.

IMPORTANT

Since the overall performance of the cluster can be impacted, the LPARs that are used to set up the OpenShift Container Platform clusters must provide sufficient compute capacity. In this context, LPAR weight management, entitlements, and CPU shares on the hypervisor level play an important role.

Operating system requirements

- One LPAR running on RHEL 8.4 or later with KVM, which is managed by libvirt

On your RHEL KVM host, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines

- Two guest virtual machines for OpenShift Container Platform compute machines
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine

5.4.6. Minimum resource requirements

Each cluster virtual machine must meet the following minimum requirements:

Virtual Machine	Operating System	vCPU [1]	Virtual RAM	Storage	IOPS
Bootstrap	RHCOS	4	16 GB	100 GB	N/A
Control plane	RHCOS	4	16 GB	100 GB	N/A
Compute	RHCOS	2	8 GB	100 GB	N/A

1. One physical core (IFL) provides two logical cores (threads) when SMT-2 is enabled. The hypervisor can provide two or more vCPUs.

5.4.7. Preferred IBM Z system environment

Hardware requirements

- Three LPARS that each have the equivalent of six IFLs, which are SMT2 enabled, for each cluster.
- Two network connections to both connect to the **LoadBalancer** service and to serve data for traffic outside the cluster.

Operating system requirements

- For high availability, two or three LPARs running on RHEL 8.4 or later with KVM, which are managed by libvirt.

On your RHEL KVM host, set up:

- Three guest virtual machines for OpenShift Container Platform control plane machines, distributed across the RHEL KVM host machines.
- At least six guest virtual machines for OpenShift Container Platform compute machines, distributed across the RHEL KVM host machines.
- One guest virtual machine for the temporary OpenShift Container Platform bootstrap machine.
- To ensure the availability of integral components in an overcommitted environment, increase the priority of the control plane by using **cpu_shares**. Do the same for infrastructure nodes, if they exist. See schedinfo in IBM Documentation.

5.4.8. Preferred resource requirements

The preferred requirements for each cluster virtual machine are:

Virtual Machine	Operating System	vCPU	Virtual RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	8	16 GB	120 GB
Compute	RHCOS	6	8 GB	120 GB

5.4.9. 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

- Recommended host practices for IBM Z & IBM® LinuxONE environments

5.4.10. 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.

5.4.10.1. 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.

5.4.10.2. 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 5.2. 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**.

Protocol	Port	Description
TCP/UDP	**30000-32767**	Kubernetes node port
ESP	N/A	IPsec Encapsulating Security Payload (ESP)

Table 5.3. Ports used for all-machine to control plane communications

Protocol	Port	Description
TCP	**6443**	Kubernetes API

Table 5.4. 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

5.4.11. 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.

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 5.5. Required DNS records

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

NOTE

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

TIP

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

5.4.11.1. 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 5.1. 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 5.2. 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.

5.4.12. 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 5.6. 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 5.7. Application Ingress load balancer

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

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

NOTE

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

5.4.12.1. 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 5.3. 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.

5.5. 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. Choose to perform either a fast track installation of Red Hat Enterprise Linux CoreOS (RHCOS) or a full installation of Red Hat Enterprise Linux CoreOS (RHCOS). For the full installation, you must set up an HTTP or HTTPS server to provide Ignition files and install images to the cluster nodes. For the fast track installation an HTTP or HTTPS server is not required, however, a DHCP server is required. See sections “Fast-track installation: Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines" and “Full installation: Creating Red Hat Enterprise Linux CoreOS (RHCOS) machines".

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

5.6. 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.

5.7. 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.

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.

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

Procedure

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

```
$ mkdir <installation_directory>
```

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

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

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

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

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

Additional resources

- Installation configuration parameters for IBM Z

5.8.1. Sample install-config.yaml file for IBM Z

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)
  architecture: s390x
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
  architecture: s390x
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-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  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 available on your OpenShift Container Platform nodes, the **hyperthreading** parameter has no effect.

IMPORTANT

If you disable **hyperthreading**, whether on your OpenShift Container Platform nodes 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 IBM Z infrastructure.

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 Installing the system in 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 Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry. The certificate file can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

18 Provide the **imageContentSources** section from the output of the command to mirror the repository.

5.8.2. Configuring the cluster-wide proxy during installation

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

Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of

them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

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

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

Procedure

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

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

5.8.3. Configuring a three-node cluster

Optionally, you can deploy zero compute machines in a minimal three node 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.

NOTE

The preferred resource for control plane nodes is six vCPUs and 21 GB. For three control plane nodes this is the memory + vCPU equivalent of a minimum five-node cluster. You should back the three nodes, each installed on a 120 GB disk, with three IFLs that are SMT2 enabled. The minimum tested setup is three vCPUs and 10 GB on a 120 GB disk for each control plane node.

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.

5.9. 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**.

5.9.1. Cluster Network Operator configuration object

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

Table 5.8. Cluster Network Operator configuration object

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

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.

defaultNetwork object configuration

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

Table 5.9. defaultNetwork object

Field	Type	Description

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 5.10. **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.
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**. This value cannot be changed after cluster installation.

Field	Type	Description
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**.

Example OpenShift SDN configuration

```
defaultNetwork:
  type: OpenShiftSDN
  openshiftSDNConfig:
    mode: NetworkPolicy
    mtu: 1450
    vxlanPort: 4789
```

Configuration for the OVN-Kubernetes network plugin

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

Table 5.11. **ovnKubernetesConfig** object

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.

Field	Type	Description
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 5.12. **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 5.13. **gatewayConfig** object

Field	Type	Description
routingViaHost	**boolean**	Set this field to **true** to send egress traffic from pods to the host networking stack. 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 5.14. **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`

5.10. 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.

NOTE

The installation program that generates the manifest and Ignition files is architecture specific and can be obtained from the client image mirror. The Linux version of the installation program runs on s390x only. This installer program is also available as a Mac OS version.

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
```

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

To install OpenShift Container Platform on IBM Z infrastructure that you provision, you must install Red Hat Enterprise Linux CoreOS (RHCOS) as Red Hat Enterprise Linux (RHEL) guest virtual 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.

You can perform a fast-track installation of RHCOS that uses a prepackaged QEMU copy-on-write (QCOW2) disk image. Alternatively, you can perform a full installation on a new QCOW2 disk image.

To add further security to your system, you can optionally install RHCOS using IBM Secure Execution before proceeding to the fast-track installation.

5.11.1. Installing RHCOS using IBM Secure Execution

Before you install RHCOS using IBM Secure Execution, you must prepare the underlying infrastructure.

IMPORTANT

Installing RHCOS using IBM Secure Execution is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

Prerequisites

- IBM z15 or later, or IBM® LinuxONE III or later.

- Red Hat Enterprise Linux (RHEL) 8 or later.
- You have a bootstrap Ignition file. The file is not protected, enabling others to view and edit it.
- You have verified that the boot image has not been altered after installation.
- You must run all your nodes as IBM Secure Execution guests.

Procedure

1. Prepare your RHEL KVM host to support IBM Secure Execution.
   - By default, KVM hosts do not support guests in IBM Secure Execution mode. To support guests in IBM Secure Execution mode, KVM hosts must boot in LPAR mode with the kernel parameter specification **prot_virt=1**. To enable **prot_virt=1** on RHEL 8, follow these steps:
     a. Navigate to **/boot/loader/entries/** to modify your bootloader configuration file ***.conf**.
     b. Add the kernel command line parameter **prot_virt=1**.
     c. Run the **zipl** command and reboot your system.
        KVM hosts that successfully start with support for IBM Secure Execution for Linux issue the following kernel message:

        ```
        prot_virt: Reserving <amount>MB as ultravisor base storage.
        ```

     d. To verify that the KVM host now supports IBM Secure Execution, run the following command:

        ```
        # cat /sys/firmware/uv/prot_virt_host
        ```

        #### Example output

        ```
        1
        ```

        The value of this attribute is 1 for Linux instances that detect their environment as consistent with that of a secure host. For other instances, the value is 0.

2. Add your host keys to the KVM guest via Ignition.
   During the first boot, RHCOS looks for your host keys to re-encrypt itself with them. RHCOS searches for files starting with **ibm-z-hostkey-** in the **/etc/se-hostkeys** directory. All host keys, for each machine the cluster is running on, must be loaded into the directory by the administrator. After first boot, you cannot run the VM on any other machines.

   > **NOTE**
   >
   > You need to prepare your Ignition file on a safe system. For example, another IBM Secure Execution guest.

   For example:

   ```
   {
     "ignition": { "version": "3.0.0" },
     "storage": {
   ```

````
    "files": [
     {
      "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
      "contents": {
       "source": "data:;base64,<base64 encoded hostkey document>"
      },
      "mode": 420
     },
     {
      "path": "/etc/se-hostkeys/ibm-z-hostkey-<your-hostkey>.crt",
      "contents": {
       "source": "data:;base64,<base64 encoded hostkey document>"
      },
      "mode": 420
     }
    ]
  }
}
```
````

NOTE

You can add as many host keys as required if you want your node to be able to run on multiple IBM Z machines.

3. To generate the Base64 encoded string, run the following command:

```
base64 <your-hostkey>.crt
```

Compared to guests not running IBM Secure Execution, the first boot of the machine is longer because the entire image is encrypted with a randomly generated LUKS passphrase before the Ignition phase.

4. Follow the fast-track installation procedure to install nodes using the IBM Secure Exection QCOW image.

Additional resources

- Introducing IBM Secure Execution for Linux
- Linux as an IBM Secure Execution host or guest

5.11.2. Fast-track installation by using a prepackaged QCOW2 disk image

Complete the following steps to create the machines in a fast-track installation of Red Hat Enterprise Linux CoreOS (RHCOS), importing a prepackaged Red Hat Enterprise Linux CoreOS (RHCOS) QEMU copy-on-write (QCOW2) disk image.

Prerequisites

- At least one LPAR running on RHEL 8.4 or later with KVM, referred to as RHEL KVM host in this procedure.
- The KVM/QEMU hypervisor is installed on the RHEL KVM host.
```

- A domain name server (DNS) that can perform hostname and reverse lookup for the nodes.
- A DHCP server that provides IP addresses.

Procedure

1. Obtain the RHEL QEMU copy-on-write (QCOW2) disk image file from the Product Downloads page on the Red Hat Customer Portal or from the RHCOS image mirror page.

   IMPORTANT

   The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Only use the appropriate RHCOS QCOW2 image described in the following procedure.

2. Download the QCOW2 disk image and Ignition files to a common directory on the RHEL KVM host.
   For example: **/var/lib/libvirt/images**

   NOTE

   The Ignition files are generated by the OpenShift Container Platform installer.

3. Create a new disk image with the QCOW2 disk image backing file for each KVM guest node.

```
$ qemu-img create -f qcow2 -F qcow2 -b /var/lib/libvirt/images/{source_rhcos_qemu}
/var/lib/libvirt/images/{vmname}.qcow2 {size}
```

4. Create the new KVM guest nodes using the Ignition file and the new disk image.

```
$ virt-install --noautoconsole \
   --connect qemu:///system \
   --name {vn_name} \
   --memory {memory} \
   --vcpus {vcpus} \
   --disk {disk} \
   --import \
   --network network={network},mac={mac} \
   --disk path={ign_file},format=raw,readonly=on,serial=ignition,startup_policy=optional
```

5.11.3. Full installation on a new QCOW2 disk image

Complete the following steps to create the machines in a full installation on a new QEMU copy-on-write (QCOW2) disk image.

Prerequisites

- At least one LPAR running on RHEL 8.4 or later with KVM, referred to as RHEL KVM host in this procedure.
- The KVM/QEMU hypervisor is installed on the RHEL KVM host.

- A domain name server (DNS) that can perform hostname and reverse lookup for the nodes.
- An HTTP or HTTPS server is set up.

Procedure

1. Obtain the RHEL kernel, initramfs, and rootfs files from the Product Downloads page on the Red Hat Customer Portal or from the RHCOS image mirror page.

   IMPORTANT

   The RHCOS images might not change with every release of OpenShift Container Platform. You must download images with the highest version that is less than or equal to the OpenShift Container Platform version that you install. Only use the appropriate RHCOS QCOW2 image described in the following procedure.

   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**

2. Move the downloaded RHEL live kernel, initramfs, and rootfs as well as the Ignition files to an HTTP or HTTPS server before you launch **virt-install**.

   NOTE

   The Ignition files are generated by the OpenShift Container Platform installer.

3. Create the new KVM guest nodes using the RHEL kernel, initramfs, and Ignition files, the new disk image, and adjusted parm line arguments.

   - For **--location**, specify the location of the kernel/initrd on the HTTP or HTTPS server.
   - For **coreos.inst.ignition_url=**, specify the Ignition file for the machine role. Use **bootstrap.ign**, **master.ign**, or **worker.ign**. Only HTTP and HTTPS protocols are supported.
   - For **coreos.live.rootfs_url=**, specify the matching rootfs artifact for the kernel and initramfs you are booting. Only HTTP and HTTPS protocols are supported.

```
$ virt-install \
   --connect qemu:///system \
   --name {vn_name} \
   --vcpus {vcpus} \
   --memory {memory_mb} \
   --disk {vn_name}.qcow2,size={image_size| default(10,true)} \
   --network network={virt_network_parm} \
   --boot hd \
   --location {media_location},kernel={rhcos_kernel},initrd={rhcos_initrd} \
   --extra-args "rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=vda
coreos.live.rootfs_url={rhcos_liveos} ip={ip}::{default_gateway}:{subnet_mask_length}:
```

```
{vn_name}:enc1:none:{MTU} nameserver={dns} coreos.inst.ignition_url={rhcos_ign}" \
  --noautoconsole \
  --wait
```

5.11.4. Advanced RHCOS installation reference

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

5.11.4.1. Networking 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 on your RHCOS nodes for ISO installations. The examples describe how to use the **ip=** and **nameserver=** kernel arguments.

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

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
```

5.12. 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.

5.13. 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

Example output

```
system:admin
```

5.14. 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.

5.15. 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.

5.15.1. 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.

5.15.2. 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.

5.15.2.1. Configuring registry storage for IBM Z

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 on IBM Z.
- 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
```

5.15.2.2. 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.

5.16. 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.

Additional resources

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH.

5.17. NEXT STEPS

- Customize your cluster.
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.
- If necessary, you can opt out of remote health reporting .
- If necessary, see Registering your disconnected cluster

CHAPTER 6. INSTALLATION CONFIGURATION PARAMETERS FOR IBM Z® AND IBM(R) LINUXONE

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

While this document refers only to IBM Z, all information in it also applies to IBM® LinuxONE.

6.1. INSTALLATION CONFIGURATION PARAMETERS

Before you deploy an OpenShift Container Platform cluster, you provide parameter values to describe your account on the cloud platform that hosts your cluster and optionally customize your cluster's platform. When you create the **install-config.yaml** installation configuration file, you provide values for the required parameters through the command line. If you customize your cluster, you can modify the **install-config.yaml** file to provide more details about the platform.

NOTE

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

6.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 6.1. 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, hyphens (**-**), and periods (**.**), 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"`
		`}`
		`}`
		`}`

6.1.2. 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.

Only IPv4 addresses are supported.

> 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 6.2. 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`

networking.clusterNetwork.cidr	Required if you use **networking.clusterNetwork**. An IP address block. An IPv4 network.	An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**.
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. The default value is **23**.

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.
		An array with an IP address block in CIDR format. For example:
		
		`networking:`
		`  serviceNetwork:`
		`   - 172.30.0.0/16`

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

6.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 6.3. 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

Parameter	Description	Values
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** (the default).	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**

Parameter	Description	Values
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** (the default).	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**

Parameter	Description	Values
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.
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 (**""**).
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	**false** or **true**

Parameter	Description	Values
	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 [Installing the system in 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.	
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.

Parameter	Description	Values
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..**.