



OpenShift Container Platform 4.18

Installing an on-premise cluster with the Agent-based Installer

Installing an on-premise OpenShift Container Platform cluster with the Agent-based
Installer

OpenShift Container Platform 4.18 Installing an on-premise cluster with the Agent-based Installer

Installing an on-premise OpenShift Container Platform cluster with the Agent-based Installer

Legal Notice

Copyright © 2025 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document describes how to install an on-premise OpenShift Container Platform cluster with the Agent-based Installer.

Table of Contents

CHAPTER 1. PREPARING TO INSTALL WITH THE AGENT-BASED INSTALLER	4
1.1. ABOUT THE AGENT-BASED INSTALLER	4
1.2. UNDERSTANDING AGENT-BASED INSTALLER	4
1.2.1. Agent-based Installer workflow	5
1.2.2. Recommended resources for topologies	6
1.3. ABOUT FIPS COMPLIANCE	7
1.4. CONFIGURING FIPS THROUGH THE AGENT-BASED INSTALLER	8
1.5. HOST CONFIGURATION	9
1.5.1. Host roles	9
1.5.2. About root device hints	10
1.6. ABOUT NETWORKING	11
1.6.1. DHCP	11
1.6.2. Static networking	11
1.7. REQUIREMENTS FOR A CLUSTER USING THE PLATFORM "NONE" OPTION	13
1.7.1. Platform "none" DNS requirements	14
1.7.1.1. Example DNS configuration for platform "none" clusters	15
1.7.2. Platform "none" Load balancing requirements	18
1.7.2.1. Example load balancer configuration for platform "none" clusters	19
1.8. EXAMPLE: BONDS AND VLAN INTERFACE NODE NETWORK CONFIGURATION	21
1.9. EXAMPLE: BONDS AND SR-IOV DUAL-NIC NODE NETWORK CONFIGURATION	23
1.10. SAMPLE INSTALL-CONFIG.YAML FILE FOR BARE METAL	25
1.11. VALIDATION CHECKS BEFORE AGENT ISO CREATION	28
1.11.1. ZTP manifests	28
1.12. NEXT STEPS	28
CHAPTER 2. UNDERSTANDING DISCONNECTED INSTALLATION MIRRORING	29
2.1. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION THROUGH THE AGENT-BASED INSTALLER	29
2.2. ABOUT MIRRORING THE OPENSIFT CONTAINER PLATFORM IMAGE REPOSITORY FOR A DISCONNECTED REGISTRY	29
2.2.1. Configuring the Agent-based Installer to use mirrored images	30
2.3. ADDITIONAL RESOURCES	31
CHAPTER 3. INSTALLING A CLUSTER	32
3.1. PREREQUISITES	32
3.2. INSTALLING OPENSIFT CONTAINER PLATFORM WITH THE AGENT-BASED INSTALLER	32
3.2.1. Downloading the Agent-based Installer	32
3.2.2. Creating the configuration inputs	32
3.2.3. Creating and booting the agent image	35
3.2.4. Verifying that the current installation host can pull release images	35
3.2.5. Tracking and verifying installation progress	37
3.3. GATHERING LOG DATA FROM A FAILED AGENT-BASED INSTALLATION	38
CHAPTER 4. INSTALLING A CLUSTER WITH CUSTOMIZATIONS	40
4.1. PREREQUISITES	40
4.2. INSTALLING OPENSIFT CONTAINER PLATFORM WITH THE AGENT-BASED INSTALLER	40
4.2.1. Downloading the Agent-based Installer	40
4.2.2. Verifying the supported architecture for an Agent-based installation	40
4.2.3. Creating the preferred configuration inputs	41
4.2.4. Creating additional manifest files	45
4.2.4.1. Creating a directory to contain additional manifests	45
4.2.4.2. Disk partitioning	45

4.2.5. Using ZTP manifests	47
4.2.6. Encrypting the disk	48
4.2.7. Creating and booting the agent image	49
4.2.8. Adding IBM Z agents with RHEL KVM	50
4.2.9. Verifying that the current installation host can pull release images	51
4.2.10. Tracking and verifying installation progress	53
4.3. SAMPLE GITOPS ZTP CUSTOM RESOURCES	54
4.4. GATHERING LOG DATA FROM A FAILED AGENT-BASED INSTALLATION	57
CHAPTER 5. PREPARING PXE ASSETS FOR OPENSIFT CONTAINER PLATFORM	59
5.1. PREREQUISITES	59
5.2. DOWNLOADING THE AGENT-BASED INSTALLER	59
5.3. CREATING THE PREFERRED CONFIGURATION INPUTS	59
5.4. CREATING THE PXE ASSETS	63
5.5. MANUALLY ADDING IBM Z AGENTS	64
5.5.1. Networking requirements for IBM Z	64
5.5.2. Configuring network overrides in an IBM Z environment	65
5.5.3. Adding IBM Z agents with z/VM	66
5.5.4. Adding IBM Z agents with RHEL KVM	67
5.5.5. Adding IBM Z agents in a Logical Partition (LPAR)	69
5.5.6. Adding IBM Z agents in a Logical Partition (LPAR) to an existing cluster	70
CHAPTER 6. PREPARING INSTALLATION ASSETS FOR ISCSI BOOTING	73
6.1. REQUIREMENTS FOR ISCSI BOOTING	73
6.2. PREREQUISITES	73
6.3. DOWNLOADING THE AGENT-BASED INSTALLER	73
6.4. CREATING THE PREFERRED CONFIGURATION INPUTS	74
6.5. CREATING THE INSTALLATION FILES	78
CHAPTER 7. PREPARING AN AGENT-BASED INSTALLED CLUSTER FOR THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR	79
7.1. PREREQUISITES	79
7.2. PREPARING AN AGENT-BASED CLUSTER DEPLOYMENT FOR THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR WHILE DISCONNECTED	79
7.3. PREPARING AN AGENT-BASED CLUSTER DEPLOYMENT FOR THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR WHILE CONNECTED	81
CHAPTER 8. INSTALLATION CONFIGURATION PARAMETERS FOR THE AGENT-BASED INSTALLER ...	87
8.1. AVAILABLE INSTALLATION CONFIGURATION PARAMETERS	87
8.1.1. Required configuration parameters	87
8.1.2. Network configuration parameters	88
8.1.3. Optional configuration parameters	91
8.1.4. Additional bare metal configuration parameters for the Agent-based Installer	97
8.1.5. Additional VMware vSphere configuration parameters	100
8.1.6. Deprecated VMware vSphere configuration parameters	103
8.2. AVAILABLE AGENT CONFIGURATION PARAMETERS	105
8.2.1. Required configuration parameters	105
8.2.2. Optional configuration parameters	106

CHAPTER 1. PREPARING TO INSTALL WITH THE AGENT-BASED INSTALLER

1.1. ABOUT THE AGENT-BASED INSTALLER

The Agent-based installation method provides the flexibility to boot your on-premise servers in any way that you choose. It combines the ease of use of the Assisted Installation service with the ability to run offline, including in air-gapped environments. Agent-based installation is a subcommand of the OpenShift Container Platform installer. It generates a bootable ISO image containing all of the information required to deploy an OpenShift Container Platform cluster, with an available release image.

The configuration is in the same format as for the installer-provisioned infrastructure and user-provisioned infrastructure installation methods. The Agent-based Installer can also optionally generate or accept Zero Touch Provisioning (ZTP) custom resources. ZTP allows you to provision new edge sites with declarative configurations of bare-metal equipment.

Table 1.1. Agent-based Installer supported architectures

CPU architecture	Connected installation	Disconnected installation
64-bit x86	✓	✓
64-bit ARM	✓	✓
ppc64le	✓	✓
s390x	✓	✓

1.2. UNDERSTANDING AGENT-BASED INSTALLER

As an OpenShift Container Platform user, you can leverage the advantages of the Assisted Installer hosted service in disconnected environments.

The Agent-based installation comprises a bootable ISO that contains the Assisted discovery agent and the Assisted Service. Both are required to perform the cluster installation, but the latter runs on only one of the hosts.



NOTE

Currently, ISO boot support on IBM Z® (**s390x**) is available only for Red Hat Enterprise Linux (RHEL) KVM, which provides the flexibility to choose either PXE or ISO-based installation. For installations with z/VM and Logical Partition (LPAR), only PXE boot is supported.

The **openshift-install agent create image** subcommand generates an ephemeral ISO based on the inputs that you provide. You can choose to provide inputs through the following manifests:

Preferred:

- **install-config.yaml**

- **agent-config.yaml**

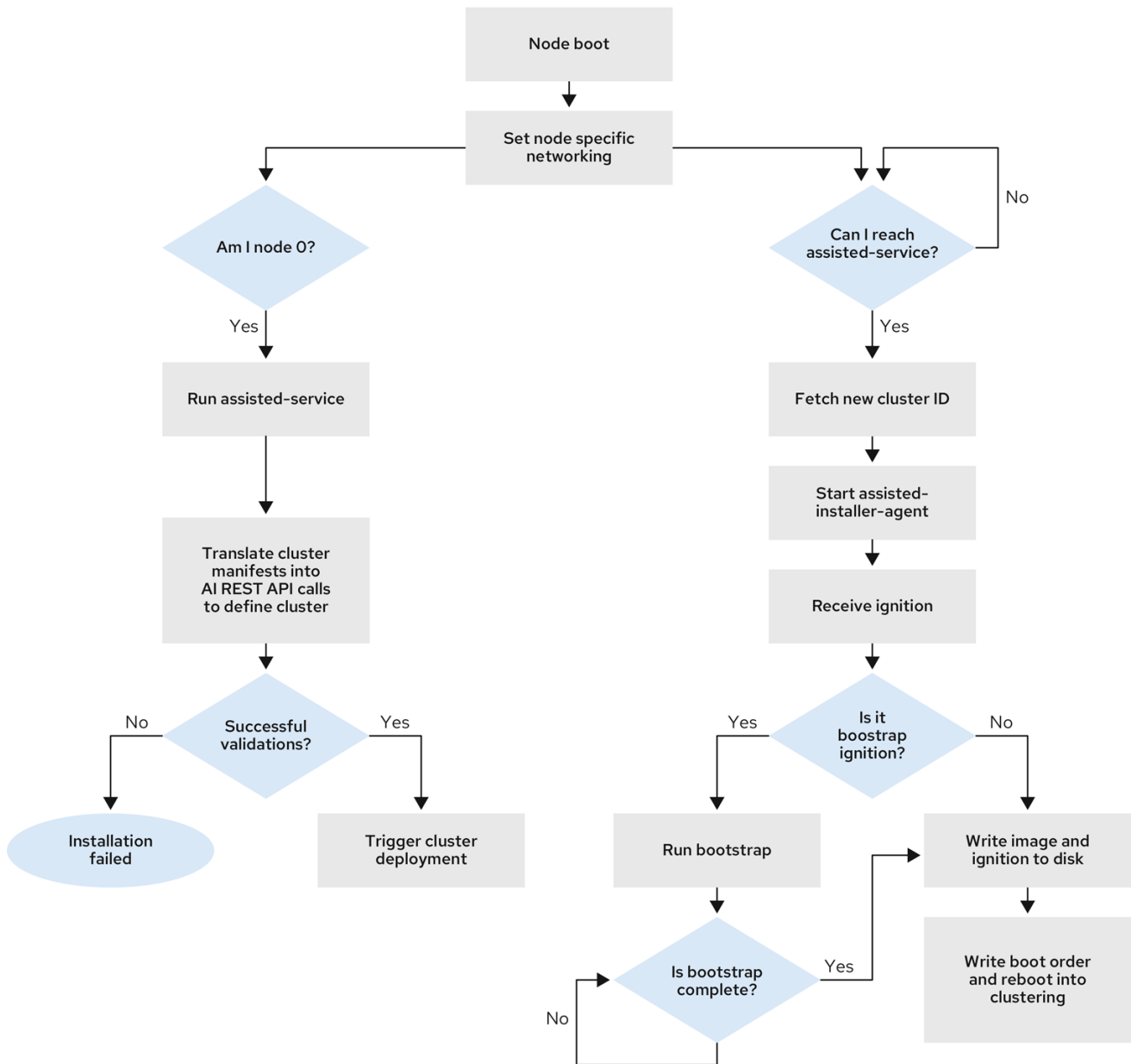
Optional: ZTP manifests

- **cluster-manifests/cluster-deployment.yaml**
- **cluster-manifests/agent-cluster-install.yaml**
- **cluster-manifests/pull-secret.yaml**
- **cluster-manifests/infraenv.yaml**
- **cluster-manifests/cluster-image-set.yaml**
- **cluster-manifests/nmstateconfig.yaml**
- **mirror/registries.conf**
- **mirror/ca-bundle.crt**

1.2.1. Agent-based Installer workflow

One of the control plane hosts runs the Assisted Service at the start of the boot process and eventually becomes the bootstrap host. This node is called the **rendezvous host** (node 0). The Assisted Service ensures that all the hosts meet the requirements and triggers an OpenShift Container Platform cluster deployment. All the nodes have the Red Hat Enterprise Linux CoreOS (RHCOS) image written to the disk. The non-bootstrap nodes reboot and initiate a cluster deployment. Once the nodes are rebooted, the rendezvous host reboots and joins the cluster. The bootstrapping is complete and the cluster is deployed.

Figure 1.1. Node installation workflow



281_OpenShift_1022

You can install a disconnected OpenShift Container Platform cluster through the **openshift-install agent create image** subcommand for the following topologies:

- **A single-node OpenShift Container Platform cluster (SNO)** A node that is both a master and worker.
- **A three-node OpenShift Container Platform cluster:** A compact cluster that has three master nodes that are also worker nodes.
- **Highly available OpenShift Container Platform cluster (HA)** Three master nodes with any number of worker nodes.

1.2.2. Recommended resources for topologies

Recommended cluster resources for the following topologies:

Table 1.2. Recommended cluster resources

Topology	Number of control plane nodes	Number of compute nodes	vCPU	Memory	Storage
Single-node cluster	1	0	8 vCPUs	16 GB of RAM	120 GB
Compact cluster	3	0 or 1	8 vCPUs	16 GB of RAM	120 GB
HA cluster	3 to 5	2 and above	8 vCPUs	16 GB of RAM	120 GB

In the **install-config.yaml**, specify the platform on which to perform the installation. The following platforms are supported:

- **baremetal**
- **vsphere**
- **external**
- **none**



IMPORTANT

For platform **none**:

- The **none** option requires the provision of DNS name resolution and load balancing infrastructure in your cluster. See *Requirements for a cluster using the platform "none" option* in the "Additional resources" section for more information.
- Review the information in the [guidelines for deploying OpenShift Container Platform on non-tested platforms](#) before you attempt to install an OpenShift Container Platform cluster in virtualized or cloud environments.

Additional resources

- [Requirements for a cluster using the platform "none" option](#)
- [Increase the network MTU](#)
- [Adding worker nodes to single-node OpenShift clusters](#)

1.3. ABOUT FIPS COMPLIANCE

For many OpenShift Container Platform customers, regulatory readiness, or compliance, on some level is required before any systems can be put into production. That regulatory readiness can be imposed by national standards, industry standards or the organization's corporate governance framework. Federal

Information Processing Standards (FIPS) compliance is one of the most critical components required in highly secure environments to ensure that only supported cryptographic technologies are allowed on nodes.



IMPORTANT

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

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

1.4. CONFIGURING FIPS THROUGH THE AGENT-BASED INSTALLER

During a cluster deployment, the Federal Information Processing Standards (FIPS) change is applied when the Red Hat Enterprise Linux CoreOS (RHCOS) machines are deployed in your cluster. For Red Hat Enterprise Linux (RHEL) machines, you must enable FIPS mode when you install the operating system on the machines that you plan to use as worker machines.



IMPORTANT

OpenShift Container Platform requires the use of a FIPS-capable installation binary to install a cluster in FIPS mode.

You can enable FIPS mode through the preferred method of **install-config.yaml** and **agent-config.yaml**:

1. You must set value of the **fips** field to **true** in the **install-config.yaml** file:

Sample install-config.yaml file

```
apiVersion: v1
baseDomain: test.example.com
metadata:
  name: sno-cluster
fips: true
```



IMPORTANT

To enable FIPS mode on IBM Z® clusters, you must also enable FIPS in either the **.parm** file or using **virt-install** as outlined in the procedures for manually adding IBM Z® agents.

2. Optional: If you are using the GitOps ZTP manifests, you must set the value of **fips** as **true** in the **agent-install.openshift.io/install-config-overrides** field in the **agent-cluster-install.yaml** file:

Sample agent-cluster-install.yaml file

```
apiVersion: extensions.hive.openshift.io/v1beta1
kind: AgentClusterInstall
```

```

metadata:
  annotations:
    agent-install.openshift.io/install-config-overrides: '{"fips": true}'
  name: sno-cluster
  namespace: sno-cluster-test

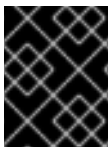
```

Additional resources

- [OpenShift Security Guide Book](#)
- [Support for FIPS cryptography](#)

1.5. HOST CONFIGURATION

You can make additional configurations for each host on the cluster in the **agent-config.yaml** file, such as network configurations and root device hints.



IMPORTANT

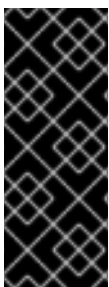
For each host you configure, you must provide the MAC address of an interface on the host to specify which host you are configuring.

1.5.1. Host roles

Each host in the cluster is assigned a role of either **master** or **worker**. You can define the role for each host in the **agent-config.yaml** file by using the **role** parameter. If you do not assign a role to the hosts, the roles will be assigned at random during installation.

It is recommended to explicitly define roles for your hosts.

The **rendezvousIP** must be assigned to a host with the **master** role. This can be done manually or by allowing the Agent-based Installer to assign the role.



IMPORTANT

You do not need to explicitly define the **master** role for the rendezvous host, however you cannot create configurations that conflict with this assignment.

For example, if you have 4 hosts with 3 of the hosts explicitly defined to have the **master** role, the last host that is automatically assigned the **worker** role during installation cannot be configured as the rendezvous host.

Sample agent-config.yaml file

```

apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: example-cluster
rendezvousIP: 192.168.111.80
hosts:
  - hostname: master-1
    role: master
    interfaces:

```

```


- name: eno1
  macAddress: 00:ef:44:21:e6:a5
- hostname: master-2
  role: master
  interfaces:
    - name: eno1
      macAddress: 00:ef:44:21:e6:a6
- hostname: master-3
  role: master
  interfaces:
    - name: eno1
      macAddress: 00:ef:44:21:e6:a7
- hostname: worker-1
  role: worker
  interfaces:
    - name: eno1
      macAddress: 00:ef:44:21:e6:a8

```

1.5.2. About root device hints

The **rootDeviceHints** parameter enables the installer to provision the Red Hat Enterprise Linux CoreOS (RHCOS) image to a particular device. The installer examines the devices in the order it discovers them, and compares the discovered values with the hint values. The installer uses the first discovered device that matches the hint value. The configuration can combine multiple hints, but a device must match all hints for the installer to select it.

Table 1.3. Subfields

Subfield	Description
deviceName	<p>A string containing a Linux device name such as /dev/vda or /dev/disk/by-path/.</p> <div>  <div> <p>NOTE</p> <p>It is recommended to use the /dev/disk/by-path/<device_path> link to the storage location.</p> </div> </div> <p>The hint must match the actual value exactly.</p>
hctl	A string containing a SCSI bus address like 0:0:0:0 . The hint must match the actual value exactly.
model	A string containing a vendor-specific device identifier. The hint can be a substring of the actual value.
vendor	A string containing the name of the vendor or manufacturer of the device. The hint can be a sub-string of the actual value.
serialNumber	A string containing the device serial number. The hint must match the actual value exactly.

Subfield	Description
minSizeGigabytes	An integer representing the minimum size of the device in gigabytes.
wwn	A string containing the unique storage identifier. The hint must match the actual value exactly. If you use the udevadm command to retrieve the wwn value, and the command outputs a value for ID_WWN_WITH_EXTENSION , then you must use this value to specify the wwn subfield.
rotational	A boolean indicating whether the device should be a rotating disk (true) or not (false).

Example usage

```
- name: master-0
  role: master
  rootDeviceHints:
    deviceName: "/dev/sda"
```

1.6. ABOUT NETWORKING

The **rendezvous IP** must be known at the time of generating the agent ISO, so that during the initial boot all the hosts can check in to the assisted service. If the IP addresses are assigned using a Dynamic Host Configuration Protocol (DHCP) server, then the **rendezvousIP** field must be set to an IP address of one of the hosts that will become part of the deployed control plane. In an environment without a DHCP server, you can define IP addresses statically.

In addition to static IP addresses, you can apply any network configuration that is in NMState format. This includes VLANs and NIC bonds.

1.6.1. DHCP

Preferred method: **install-config.yaml** and **agent-config.yaml**

You must specify the value for the **rendezvousIP** field. The **networkConfig** fields can be left blank:

Sample agent-config.yaml file

```
apiVersion: v1alpha1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 1
```

1 The IP address for the rendezvous host.

1.6.2. Static networking

- Preferred method: **install-config.yaml** and **agent-config.yaml**

Sample agent-config.yaml file

```

cat > agent-config.yaml << EOF
apiVersion: v1alpha1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 ❶
hosts:
  - hostname: master-0
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a5 ❷
    networkConfig:
      interfaces:
        - name: eno1
          type: ethernet
          state: up
          mac-address: 00:ef:44:21:e6:a5
          ipv4:
            enabled: true
            address:
              - ip: 192.168.111.80 ❸
                prefix-length: 23 ❹
            dhcp: false
      dns-resolver:
        config:
          server:
            - 192.168.111.1 ❺
      routes:
        config:
          - destination: 0.0.0.0/0
            next-hop-address: 192.168.111.1 ❻
            next-hop-interface: eno1
            table-id: 254
EOF

```

- ❶ If a value is not specified for the **rendezvousIP** field, one address will be chosen from the static IP addresses specified in the **networkConfig** fields.
- ❷ The MAC address of an interface on the host, used to determine which host to apply the configuration to.
- ❸ The static IP address of the target bare metal host.
- ❹ The static IP address's subnet prefix for the target bare metal host.
- ❺ The DNS server for the target bare metal host.
- ❻ Next hop address for the node traffic. This must be in the same subnet as the IP address set for the specified interface.

b. Optional method: GitOps ZTP manifests

The optional method of the GitOps ZTP custom resources comprises 6 custom resources; you can configure static IPs in the **nmstateconfig.yaml** file.

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: master-0
  namespace: openshift-machine-api
  labels:
    cluster0-nmstate-label-name: cluster0-nmstate-label-value
spec:
  config:
    interfaces:
      - name: eth0
        type: ethernet
        state: up
        mac-address: 52:54:01:aa:aa:a1
        ipv4:
          enabled: true
          address:
            - ip: 192.168.122.2 ❶
              prefix-length: 23 ❷
          dhcp: false
        dns-resolver:
          config:
            server:
              - 192.168.122.1 ❸
        routes:
          config:
            - destination: 0.0.0.0/0
              next-hop-address: 192.168.122.1 ❹
              next-hop-interface: eth0
              table-id: 254
    interfaces:
      - name: eth0
        macAddress: 52:54:01:aa:aa:a1 ❺
```

- ❶ The static IP address of the target bare metal host.
- ❷ The static IP address's subnet prefix for the target bare metal host.
- ❸ The DNS server for the target bare metal host.
- ❹ Next hop address for the node traffic. This must be in the same subnet as the IP address set for the specified interface.
- ❺ The MAC address of an interface on the host, used to determine which host to apply the configuration to.

The rendezvous IP is chosen from the static IP addresses specified in the **config** fields.

1.7. REQUIREMENTS FOR A CLUSTER USING THE PLATFORM "NONE" OPTION

This section describes the requirements for an Agent-based OpenShift Container Platform installation that is configured to use the platform **none** option.



IMPORTANT

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

1.7.1. Platform "none" 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 control plane and compute machines

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

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




NOTE

It is recommended to use a DHCP server to provide the hostnames to each cluster node.

The following DNS records are required for an OpenShift Container Platform cluster using the platform **none** option and they must be in place before installation. In each record, **<cluster_name>** is the cluster name and **<base_domain>** is the base domain that you specify in the **install-config.yaml** file. A complete DNS record takes the form: **<component>.<cluster_name>.<base_domain>..**

Table 1.4. 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>.	<p>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.</p> <div>  <p>IMPORTANT</p> <p>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.</p> </div>
Routes	*.apps.<cluster_name>.<base_domain>.	<p>A wildcard DNS A/AAAA or CNAME record that refers to the application ingress load balancer. The application ingress load balancer targets the machines that run the Ingress Controller pods. The Ingress Controller pods run on the compute machines by default. These records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.</p> <p>For example, console-openshift-console.apps.<cluster_name>.<base_domain> is used as a wildcard route to the OpenShift Container Platform console.</p>
Control plane machines	<master><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	<worker><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.

1.7.1.1. Example DNS configuration for platform "none" clusters

This section provides A and PTR record configuration samples that meet the DNS requirements for deploying OpenShift Container Platform using the platform **none** option. 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 platform "none" cluster

The following example is a BIND zone file that shows sample A records for name resolution in a cluster using the platform **none** option.

Example 1.1. Sample DNS zone database

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
IN MX 10 smtp.example.com.
;
;
ns1.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
;
master0.ocp4.example.com. IN A 192.168.1.97 ④
master1.ocp4.example.com. IN A 192.168.1.98 ⑤
master2.ocp4.example.com. IN A 192.168.1.99 ⑥
;
worker0.ocp4.example.com. IN A 192.168.1.11 ⑦
worker1.ocp4.example.com. IN A 192.168.1.7 ⑧
;
;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 5 6 Provides name resolution for the control plane machines.

7 8 Provides name resolution for the compute machines.

Example DNS PTR record configuration for a platform "none" cluster

The following example BIND zone file shows sample PTR records for reverse name resolution in a cluster using the platform **none** option.

Example 1.2. Sample DNS zone database for reverse records

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H ; refresh (3 hours)
    30M ; retry (30 minutes)
    2W ; expiry (2 weeks)
    1W ) ; minimum (1 week)
IN NS ns1.example.com.
;
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
;
97.1.168.192.in-addr.arpa. IN PTR master0.ocp4.example.com. 3
98.1.168.192.in-addr.arpa. IN PTR master1.ocp4.example.com. 4
99.1.168.192.in-addr.arpa. IN PTR master2.ocp4.example.com. 5
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 6
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com. 7
;
;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 4 5 Provides reverse DNS resolution for the control plane machines.

6 7 Provides reverse DNS resolution for the compute machines.

**NOTE**

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

1.7.2. Platform "none" Load balancing requirements

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**

These requirements do not apply to single-node OpenShift clusters using the platform **none** option.

**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, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the API routes.
 - A stateless load balancing algorithm. The options vary based on the load balancer implementation.

**IMPORTANT**

Do not configure session persistence for an API load balancer.

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

Table 1.5. API load balancer

Port	Back-end machines (pool members)	Internal	External	Description
6443	Control plane. You must configure the /readyz endpoint for the API server health check probe.	X	X	Kubernetes API server
22623	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, SSL Passthrough, or SSL Bridge mode. If you use SSL Bridge mode, you must enable Server Name Indication (SNI) for the ingress routes.
- A connection-based or session-based persistence is recommended, based on the options available and types of applications that will be hosted on the platform.

TIP

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

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

Table 1.6. 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.

1.7.2.1. Example load balancer configuration for platform "none" clusters

This section provides an example API and application Ingress load balancer configuration that meets the load balancing requirements for clusters using the platform **none** option. 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 1.3. 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
  server master0 master0.ocp4.example.com:6443 check inter 1s
  server master1 master1.ocp4.example.com:6443 check inter 1s
  server master2 master2.ocp4.example.com:6443 check inter 1s
listen machine-config-server-22623 2
  bind *:22623
  mode tcp
  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 3
  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 4
  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 Port **22623** handles the machine config server traffic and points to the control plane machines.
- 3 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.
- 4 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 -nltpu** on the HAProxy node.

1.8. EXAMPLE: BONDS AND VLAN INTERFACE NODE NETWORK CONFIGURATION

The following **agent-config.yaml** file is an example of a manifest for bond and VLAN interfaces.

```
apiVersion: v1alpha1
kind: AgentConfig
rendezvousIP: 10.10.10.14
hosts:
- hostname: master0
  role: master
  interfaces:
  - name: enp0s4
    macAddress: 00:21:50:90:c0:10
  - name: enp0s5
    macAddress: 00:21:50:90:c0:20
  networkConfig:
    interfaces:
    - name: bond0.300 1
      type: vlan 2
      state: up
      vlan:
        base-iface: bond0
        id: 300
```

```

    ipv4:
      enabled: true
      address:
        - ip: 10.10.10.14
          prefix-length: 24
      dhcp: false
  - name: bond0 ③
    type: bond ④
    state: up
    mac-address: 00:21:50:90:c0:10 ⑤
    ipv4:
      enabled: false
    ipv6:
      enabled: false
    link-aggregation:
      mode: active-backup ⑥
      options:
        miimon: "150" ⑦
      port:
        - enp0s4
        - enp0s5
    dns-resolver: ⑧
    config:
      server:
        - 10.10.10.11
        - 10.10.10.12
    routes:
      config:
        - destination: 0.0.0.0/0
          next-hop-address: 10.10.10.10 ⑨
          next-hop-interface: bond0.300 ⑩
          table-id: 254

```

- ① ③ Name of the interface.
- ② The type of interface. This example creates a VLAN.
- ④ The type of interface. This example creates a bond.
- ⑤ The mac address of the interface.
- ⑥ The **mode** attribute specifies the bonding mode.
- ⑦ Specifies the MII link monitoring frequency in milliseconds. This example inspects the bond link every 150 milliseconds.
- ⑧ Optional: Specifies the search and server settings for the DNS server.
- ⑨ Next hop address for the node traffic. This must be in the same subnet as the IP address set for the specified interface.
- ⑩ Next hop interface for the node traffic.

1.9. EXAMPLE: BONDS AND SR-IOV DUAL-NIC NODE NETWORK CONFIGURATION

The following **agent-config.yaml** file is an example of a manifest for dual port NIC with a bond and SR-IOV interfaces:

```
apiVersion: v1alpha1
kind: AgentConfig
rendezvousIP: 10.10.10.14
hosts:
- hostname: worker-1
  interfaces:
  - name: eno1
    macAddress: 0c:42:a1:55:f3:06
  - name: eno2
    macAddress: 0c:42:a1:55:f3:07
networkConfig: 1
  interfaces: 2
  - name: eno1 3
    type: ethernet 4
    state: up
    mac-address: 0c:42:a1:55:f3:06
    ipv4:
      enabled: true
      dhcp: false 5
    ethernet:
      sr-iov:
        total-vfs: 2 6
    ipv6:
      enabled: false
  - name: sriov:eno1:0
    type: ethernet
    state: up 7
    ipv4:
      enabled: false 8
    ipv6:
      enabled: false
      dhcp: false
  - name: sriov:eno1:1
    type: ethernet
    state: down
  - name: eno2
    type: ethernet
    state: up
    mac-address: 0c:42:a1:55:f3:07
    ipv4:
      enabled: true
    ethernet:
      sr-iov:
        total-vfs: 2
    ipv6:
      enabled: false
  - name: sriov:eno2:0
    type: ethernet
```

```

state: up
ipv4:
  enabled: false
ipv6:
  enabled: false
- name: sriov:eno2:1
  type: ethernet
  state: down
- name: bond0
  type: bond
  state: up
  min-tx-rate: 100 9
  max-tx-rate: 200 10
  link-aggregation:
    mode: active-backup 11
    options:
      primary: sriov:eno1:0 12
  port:
    - sriov:eno1:0
    - sriov:eno2:0
  ipv4:
    address:
      - ip: 10.19.16.57 13
      prefix-length: 23
    dhcp: false
    enabled: true
  ipv6:
    enabled: false
  dns-resolver:
    config:
      server:
        - 10.11.5.160
        - 10.2.70.215
  routes:
    config:
      - destination: 0.0.0.0/0
        next-hop-address: 10.19.17.254
        next-hop-interface: bond0 14
        table-id: 254

```

- 1** The **networkConfig** field contains information about the network configuration of the host, with subfields including **interfaces**, **dns-resolver**, and **routes**.
- 2** The **interfaces** field is an array of network interfaces defined for the host.
- 3** The name of the interface.
- 4** The type of interface. This example creates an ethernet interface.
- 5** Set this to **false** to disable DHCP for the physical function (PF) if it is not strictly required.
- 6** Set this to the number of SR-IOV virtual functions (VFs) to instantiate.
- 7** Set this to **up**.

- 8 Set this to **false** to disable IPv4 addressing for the VF attached to the bond.
- 9 Sets a minimum transmission rate, in Mbps, for the VF. This sample value sets a rate of 100 Mbps.
 - This value must be less than or equal to the maximum transmission rate.
 - Intel NICs do not support the **min-tx-rate** parameter. For more information, see [BZ#1772847](#).
- 10 Sets a maximum transmission rate, in Mbps, for the VF. This sample value sets a rate of 200 Mbps.
- 11 Sets the desired bond mode.
- 12 Sets the preferred port of the bonding interface. The primary device is the first of the bonding interfaces to be used and is not abandoned unless it fails. This setting is particularly useful when one NIC in the bonding interface is faster and, therefore, able to handle a bigger load. This setting is only valid when the bonding interface is in **active-backup** mode (mode 1) and **balance-tilb** (mode 5).
- 13 Sets a static IP address for the bond interface. This is the node IP address.
- 14 Sets **bond0** as the gateway for the default route.

Additional resources

- [Configuring network bonding](#)

1.10. SAMPLE INSTALL-CONFIG.YAML FILE FOR BARE METAL

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

```

apiVersion: v1
baseDomain: example.com 1
compute: 2
- name: worker
  replicas: 0 3
  architecture: amd64
controlPlane: 4
  name: master
  replicas: 1 5
  architecture: amd64
metadata:
  name: sno-cluster 6
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14 7
      hostPrefix: 23 8
  networkType: OVNKubernetes 9
  serviceNetwork: 10
    - 172.30.0.0/16
platform:
  none: {} 11

```

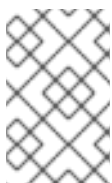
```
fips: false 12
pullSecret: '{"auths": ...}' 13
sshKey: 'ssh-ed25519 AAAA...' 14
```

- ¹ The base domain of the cluster. All DNS records must be sub-domains of this base and include the cluster name.
- ² ⁴ 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 This parameter controls the number of compute machines that the Agent-based installation waits to discover before triggering the installation process. It is the number of compute machines that must be booted with the generated ISO.

**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.

- ⁵ The number of control plane machines that you add to the cluster. Because the cluster uses these values as the number of etcd endpoints in the cluster, the value must match the number of control plane machines that you deploy.
- ⁶ The cluster name that you specified in your DNS records.
- ⁷ 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.

- ⁸ The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23**, then each node is assigned a **/23** subnet out of the given **cidr**, which allows for 510 ($2^{(32 - 23)} - 2$) pod IP addresses. If you are required to provide access to nodes from an external network, configure load balancers and routers to manage the traffic.
- ⁹ The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- ¹⁰ The IP address pool to use for service IP addresses. You can enter only one IP address pool. This block must not overlap with existing physical networks. If you need to access the services from an external network, configure load balancers and routers to manage the traffic.
- ¹¹ You must set the platform to **none** for a single-node cluster. You can set the platform to **vsphere**, **baremetal**, or **none** for multi-node clusters.



NOTE

If you set the platform to **vsphere** or **baremetal**, you can configure IP address endpoints for cluster nodes in three ways:

- IPv4
- IPv6
- IPv4 and IPv6 in parallel (dual-stack)

Example of dual-stack networking

```
networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
      - 192.168.11.3
      - 2001:DB8::4
    ingressVIPs:
      - 192.168.11.4
      - 2001:DB8::5
```

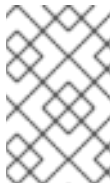
- 12** Whether to enable or disable FIPS mode. By default, FIPS mode is not enabled. If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided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



IMPORTANT

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

- 13** 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.
- 14** The SSH public key for the **core** user in Red Hat Enterprise Linux CoreOS (RHCOS).

**NOTE**

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

1.11. VALIDATION CHECKS BEFORE AGENT ISO CREATION

The Agent-based Installer performs validation checks on user defined YAML files before the ISO is created. Once the validations are successful, the agent ISO is created.

install-config.yaml

- **baremetal**, **vsphere** and **none** platforms are supported.
- The **networkType** parameter must be **OVNKubernetes** in the case of **none** platform.
- **apiVIPs** and **ingressVIPs** parameters must be set for bare metal and vSphere platforms.
- Some host-specific fields in the bare metal platform configuration that have equivalents in **agent-config.yaml** file are ignored. A warning message is logged if these fields are set.

agent-config.yaml

- Each interface must have a defined MAC address. Additionally, all interfaces must have a different MAC address.
- At least one interface must be defined for each host.
- World Wide Name (WWN) vendor extensions are not supported in root device hints.
- The **role** parameter in the **host** object must have a value of either **master** or **worker**.

1.11.1. ZTP manifests

agent-cluster-install.yaml

- For IPv6, the only supported value for the **networkType** parameter is **OVNKubernetes**. The **OpenshiftSDN** value can be used only for IPv4.

cluster-image-set.yaml

- The **ReleaseImage** parameter must match the release defined in the installer.

1.12. NEXT STEPS

- [Installing a cluster](#)
- [Installing a cluster with customizations](#)

CHAPTER 2. UNDERSTANDING DISCONNECTED INSTALLATION MIRRORING

You can use a mirror registry for disconnected installations and to ensure that your clusters only use container images that satisfy your organization's controls on external content. Before you install a cluster on infrastructure that you provision in a disconnected environment, you must mirror the required container images into that environment. To mirror container images, you must have a registry for mirroring.

2.1. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION THROUGH THE AGENT-BASED INSTALLER

You can use one of the following procedures to mirror your OpenShift Container Platform image repository to your mirror registry:

- [Mirroring images for a disconnected installation](#)
- [Mirroring images for a disconnected installation by using the oc-mirror plugin v2](#)

2.2. ABOUT MIRRORING THE OPENSIFT CONTAINER PLATFORM IMAGE REPOSITORY FOR A DISCONNECTED REGISTRY

To use mirror images for a disconnected installation with the Agent-based Installer, you must modify the **install-config.yaml** file.

You can mirror the release image by using the output of either the **oc adm release mirror** or **oc mirror** command. This is dependent on which command you used to set up the mirror registry.

The following example shows the output of the **oc adm release mirror** command.

```
$ oc adm release mirror
```

Example output

To use the new mirrored repository to install, add the following section to the **install-config.yaml**:

```
imageContentSources:

  mirrors:
    virthost.ostest.test.metalkube.org:5000/localimages/local-release-image
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
  mirrors:
    virthost.ostest.test.metalkube.org:5000/localimages/local-release-image
    source: registry.ci.openshift.org/ocp/release
```

The following example shows part of the **imageContentSourcePolicy.yaml** file generated by the **oc-mirror** plugin. The file can be found in the results directory, for example **oc-mirror-workspace/results-1682697932/**.

Example imageContentSourcePolicy.yaml file

Example registries.conf file

```
[[registry]]  
location = "registry.ci.openshift.org/ocp/release" mirror-by-digest-only = true  
  
[[registry.mirror]] location = "virthost.ostest.test.metalkube.org:5000/localimages/local-  
release-image"  
  
[[registry]]  
location = "quay.io/openshift-release-dev/ocp-v4.0-art-dev" mirror-by-digest-only = true  
  
[[registry.mirror]] location = "virthost.ostest.test.metalkube.org:5000/localimages/local-  
release-image"
```

2.3. ADDITIONAL RESOURCES

- [Installing an OpenShift Container Platform cluster with the Agent-based Installer](#)

CHAPTER 3. INSTALLING A CLUSTER

You can install a basic OpenShift Container Platform cluster using the Agent-based Installer.

For procedures that include optional customizations you can make while using the Agent-based Installer, see [Installing a cluster with customizations](#).

3.1. PREREQUISITES

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

3.2. INSTALLING OPENSIFT CONTAINER PLATFORM WITH THE AGENT-BASED INSTALLER

The following procedures deploy a single-node OpenShift Container Platform in a disconnected environment. You can use these procedures as a basis and modify according to your requirements.

3.2.1. Downloading the Agent-based Installer

Use this procedure to download the Agent-based Installer and the CLI needed for your installation.

Procedure

1. Log in to the OpenShift Container Platform web console using your login credentials.
2. Navigate to [Datacenter](#).
3. Click **Run Agent-based Installer locally**.
4. Select the operating system and architecture for the **OpenShift Installer** and **Command line interface**.
5. Click **Download Installer** to download and extract the install program.
6. Download or copy the pull secret by clicking on **Download pull secret** or **Copy pull secret**.
7. Click **Download command-line tools** and place the **openshift-install** binary in a directory that is on your **PATH**.

3.2.2. Creating the configuration inputs

You must create the configuration files that are used by the installation program to create the agent image.

Procedure

1. Place the **openshift-install** binary in a directory that is on your PATH.

2. Create a directory to store the install configuration by running the following command:

```
$ mkdir ~/<directory_name>
```

3. Create the **install-config.yaml** file by running the following command:

[illegible]

- 1** Specify the system architecture. Valid values are **amd64**, **arm64**, **ppc64le**, and **s390x**.

If you are using the release image with the **multi** payload, you can install the cluster on different architectures such as **arm64**, **amd64**, **s390x**, and **ppc64le**. Otherwise, you can install the cluster only on the **release architecture** displayed in the output of the **openshift-install version** command. For more information, see "Verifying the supported architecture for installing an Agent-based Installer cluster".

- 2 Required. Specify your cluster name.
- 3 The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- 4 Specify your platform.

**NOTE**

For bare metal platforms, host settings made in the platform section of the **install-config.yaml** file are used by default, unless they are overridden by configurations made in the **agent-config.yaml** file.

- 5 Specify your pull secret.
- 6 Specify your SSH public key.
- 7 Provide 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. You must specify this parameter if you are using a disconnected mirror registry.
- 8 Provide the **imageContentSources** section according to the output of the command that you used to mirror the repository. You must specify this parameter if you are using a disconnected mirror registry.

**IMPORTANT**

- When using the **oc adm release mirror** command, use the output from the **imageContentSources** section.
- When using the **oc mirror** command, use the **repositoryDigestMirrors** section of the **ImageContentSourcePolicy** file that results from running the command.
- The **ImageContentSourcePolicy** resource is deprecated.

4. Create the **agent-config.yaml** file by running the following command:

```
$ cat > agent-config.yaml << EOF
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: fd2e:6f44:5dd8:c956::50 1
EOF
```

- 1 This IP address is used to determine which node performs the bootstrapping process as well as running the **assisted-service** component. You must provide the rendezvous IP address when you do not specify at least one host IP address in the **networkConfig** parameter. If this address is not provided, one IP address is selected from the provided host **networkConfig** parameter.

3.2.3. Creating and booting the agent image

Use this procedure to boot the agent image on your machines.

Procedure

1. Create the agent image by running the following command:

```
$ openshift-install --dir <install_directory> agent create image
```



NOTE

Red Hat Enterprise Linux CoreOS (RHCOS) supports multipathing on the primary disk, allowing stronger resilience to hardware failure to achieve higher host availability. Multipathing is enabled by default in the agent ISO image, with a default **/etc/multipath.conf** configuration.

2. Boot the **agent.x86_64.iso**, **agent.aarch64.iso**, or **agent.s390x.iso** image on the bare metal machines.

3.2.4. Verifying that the current installation host can pull release images

After you boot the agent image and network services are made available to the host, the agent console application performs a pull check to verify that the current host can retrieve release images.

If the primary pull check passes, you can quit the application to continue with the installation. If the pull check fails, the application performs additional checks, as seen in the **Additional checks** section of the TUI, to help you troubleshoot the problem. A failure for any of the additional checks is not necessarily critical as long as the primary pull check succeeds.

If there are host network configuration issues that might cause an installation to fail, you can use the console application to make adjustments to your network configurations.



IMPORTANT

If the agent console application detects host network configuration issues, the installation workflow will be halted until the user manually stops the console application and signals the intention to proceed.

Procedure

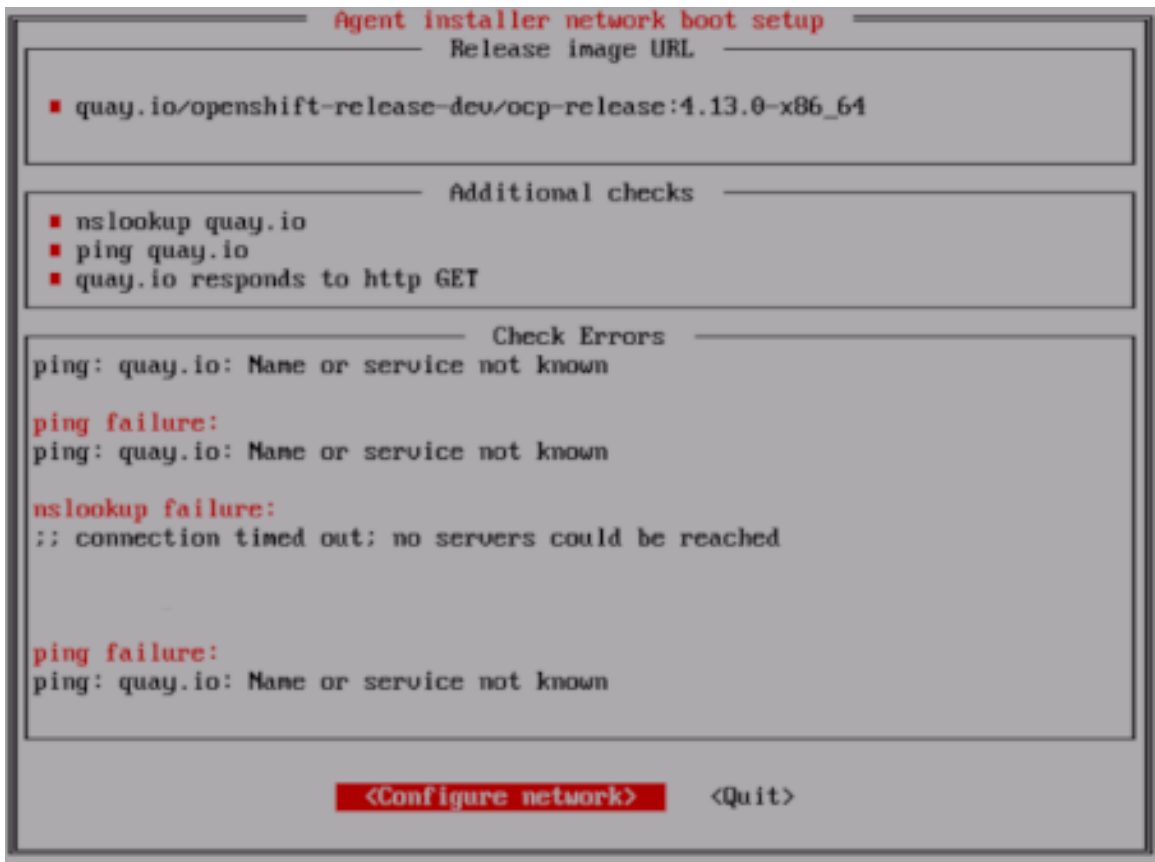
1. Wait for the agent console application to check whether or not the configured release image can be pulled from a registry.
2. If the agent console application states that the installer connectivity checks have passed, wait for the prompt to time out to continue with the installation.

**NOTE**

You can still choose to view or change network configuration settings even if the connectivity checks have passed.

However, if you choose to interact with the agent console application rather than letting it time out, you must manually quit the TUI to proceed with the installation.

3. If the agent console application checks have failed, which is indicated by a red icon beside the **Release image URL** pull check, use the following steps to reconfigure the host's network settings:
 - a. Read the **Check Errors** section of the TUI. This section displays error messages specific to the failed checks.



- b. Select **Configure network** to launch the NetworkManager TUI.
- c. Select **Edit a connection** and select the connection you want to reconfigure.
- d. Edit the configuration and select **OK** to save your changes.
- e. Select **Back** to return to the main screen of the NetworkManager TUI.
- f. Select **Activate a Connection**.
- g. Select the reconfigured network to deactivate it.
- h. Select the reconfigured network again to reactivate it.
- i. Select **Back** and then select **Quit** to return to the agent console application.

- j. Wait at least five seconds for the continuous network checks to restart using the new network configuration.
- k. If the **Release image URL** pull check succeeds and displays a green icon beside the URL, select **Quit** to exit the agent console application and continue with the installation.

3.2.5. Tracking and verifying installation progress

Use the following procedure to track installation progress and to verify a successful installation.

Prerequisites

- You have configured a DNS record for the Kubernetes API server.

Procedure

1. Optional: To know when the bootstrap host (rendezvous host) reboots, run the following command:

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

- 1 For **<install_directory>**, specify the path to the directory where the agent ISO was generated.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
.....
.....
INFO Bootstrap configMap status is complete
INFO cluster bootstrap is complete
```

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

2. To track the progress and verify successful installation, run the following command:

```
$ openshift-install --dir <install_directory> agent wait-for install-complete 1
```

- 1 For **<install_directory>** directory, specify the path to the directory where the agent ISO was generated.

Example output

```
.....
.....
INFO Cluster is installed
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run
```

```
INFO export KUBECONFIG=/home/core/installer/auth/kubeconfig
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.sno-cluster.test.example.com
```

3.3. GATHERING LOG DATA FROM A FAILED AGENT-BASED INSTALLATION

Use the following procedure to gather log data about a failed Agent-based installation to provide for a support case.

Prerequisites

- You have configured a DNS record for the Kubernetes API server.

Procedure

- Run the following command and collect the output:

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

Example error message

```
...
ERROR Bootstrap failed to complete: : bootstrap process timed out: context deadline exceeded
```

- If the output from the previous command indicates a failure, or if the bootstrap is not progressing, run the following command to connect to the rendezvous host and collect the output:

```
$ ssh core@<node-ip> agent-gather -O >agent-gather.tar.xz
```



NOTE

Red Hat Support can diagnose most issues using the data gathered from the rendezvous host, but if some hosts are not able to register, gathering this data from every host might be helpful.

- If the bootstrap completes and the cluster nodes reboot, run the following command and collect the output:

```
$ ./openshift-install --dir <install_directory> agent wait-for install-complete --log-level=debug
```

- If the output from the previous command indicates a failure, perform the following steps:
 - Export the **kubeconfig** file to your environment by running the following command:

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

- Gather information for debugging by running the following command:

■

```
$ oc adm must-gather
```

- c. Create a compressed file from the **must-gather** directory that was just created in your working directory by running the following command:

```
$ tar cvaf must-gather.tar.gz <must_gather_directory>
```

5. Excluding the **/auth** subdirectory, attach the installation directory used during the deployment to your support case on the [Red Hat Customer Portal](#).
6. Attach all other data gathered from this procedure to your support case.

CHAPTER 4. INSTALLING A CLUSTER WITH CUSTOMIZATIONS

Use the following procedures to install an OpenShift Container Platform cluster with customizations using the Agent-based Installer.

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](#).
- If you use a firewall or proxy, you [configured it to allow the sites](#) that your cluster requires access to.

4.2. INSTALLING OPENSIFT CONTAINER PLATFORM WITH THE AGENT-BASED INSTALLER

The following procedures deploy a single-node OpenShift Container Platform in a disconnected environment. You can use these procedures as a basis and modify according to your requirements.

4.2.1. Downloading the Agent-based Installer

Use this procedure to download the Agent-based Installer and the CLI needed for your installation.

Procedure

1. Log in to the OpenShift Container Platform web console using your login credentials.
2. Navigate to [Datacenter](#).
3. Click **Run Agent-based Installer locally**.
4. Select the operating system and architecture for the **OpenShift Installer** and **Command line interface**.
5. Click **Download Installer** to download and extract the install program.
6. Download or copy the pull secret by clicking on **Download pull secret** or **Copy pull secret**.
7. Click **Download command-line tools** and place the **openshift-install** binary in a directory that is on your **PATH**.

4.2.2. Verifying the supported architecture for an Agent-based installation

Before installing an OpenShift Container Platform cluster using the Agent-based Installer, you can verify the supported architecture on which you can install the cluster. This procedure is optional.

Prerequisites

- You installed the OpenShift CLI (**oc**).

- You have downloaded the installation program.

Procedure

1. Log in to the OpenShift CLI (**oc**).
2. Check your release payload by running the following command:

```
$ ./openshift-install version
```

Example output

```
./openshift-install 4.18.0
built from commit abc123def456
release image quay.io/openshift-release-dev/ocp-
release@sha256:123abc456def789ghi012jkl345mno678pqr901stu234vwx567yz0
release architecture amd64
```

If you are using the release image with the **multi** payload, the **release architecture** displayed in the output of this command is the default architecture.

3. To check the architecture of the payload, run the following command:

```
$ oc adm release info <release_image> -o jsonpath="{.metadata.metadata}" 1
```

- 1** Replace **<release_image>** with the release image. For example: **quay.io/openshift-release-dev/ocp-release@sha256:123abc456def789ghi012jkl345mno678pqr901stu234vwx567yz0**.

.Example output when the release image uses the **multi** payload

```
{"release.openshift.io architecture":"multi"}
```

If you are using the release image with the **multi** payload, you can install the cluster on different architectures such as **arm64**, **amd64**, **s390x**, and **ppc64le**. Otherwise, you can install the cluster only on the **release architecture** displayed in the output of the **openshift-install version** command.

4.2.3. Creating the preferred configuration inputs

Use this procedure to create the preferred configuration inputs used to create the agent image.



NOTE

Configuring the **install-config.yaml** and **agent-config.yaml** files is the preferred method for using the Agent-based Installer. Using GitOps ZTP manifests is optional.

Procedure

1. Install the **nmstate** dependency by running the following command:

```
$ sudo dnf install /usr/bin/nmstatectl -y
```

2. Place the **openshift-install** binary in a directory that is on your PATH.
3. Create a directory to store the install configuration by running the following command:

```
$ mkdir ~/<directory_name>
```

4. Create the **install-config.yaml** file by running the following command:

```
$ cat << EOF > ./<directory_name>/install-config.yaml
apiVersion: v1
baseDomain: test.example.com
compute:
- architecture: amd64 ❶
  hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  replicas: 1
metadata:
  name: sno-cluster ❷
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/16
  networkType: OVNKubernetes ❸
  serviceNetwork:
  - 172.30.0.0/16
platform: ❹
  none: {}
pullSecret: '<pull_secret>' ❺
sshKey: '<ssh_pub_key>' ❻
EOF
```

- ❶ Specify the system architecture. Valid values are **amd64**, **arm64**, **ppc64le**, and **s390x**.

If you are using the release image with the **multi** payload, you can install the cluster on different architectures such as **arm64**, **amd64**, **s390x**, and **ppc64le**. Otherwise, you can install the cluster only on the **release architecture** displayed in the output of the **openshift-install version** command. For more information, see "Verifying the supported architecture for installing an Agent-based Installer cluster".

- ❷ Required. Specify your cluster name.
- ❸ The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- ❹ Specify your platform.

**NOTE**

For bare-metal platforms, host settings made in the platform section of the **install-config.yaml** file are used by default, unless they are overridden by configurations made in the **agent-config.yaml** file.

- 5 Specify your pull secret.
- 6 Specify your SSH public key.

**NOTE**

If you set the platform to **vSphere** or **baremetal**, you can configure IP address endpoints for cluster nodes in three ways:

- IPv4
- IPv6
- IPv4 and IPv6 in parallel (dual-stack)

IPv6 is supported only on bare metal platforms.

Example of dual-stack networking

```
networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
      - 192.168.11.3
      - 2001:DB8::4
    ingressVIPs:
      - 192.168.11.4
      - 2001:DB8::5
```

**NOTE**

When you use a disconnected mirror registry, you must add the certificate file that you created previously for your mirror registry to the **additionalTrustBundle** field of the **install-config.yaml** file.

5. Create the **agent-config.yaml** file by running the following command:

```
$ cat > agent-config.yaml << EOF
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 ❶
hosts: ❷
- hostname: master-0 ❸
  interfaces:
    - name: eno1
      macAddress: 00:ef:44:21:e6:a5
  rootDeviceHints: ❹
    deviceName: /dev/sdb
  networkConfig: ❺
    interfaces:
      - name: eno1
        type: ethernet
        state: up
        mac-address: 00:ef:44:21:e6:a5
      ipv4:
        enabled: true
        address:
          - ip: 192.168.111.80
            prefix-length: 23
        dhcp: false
    dns-resolver:
      config:
        server:
          - 192.168.111.1
    routes:
      config:
        - destination: 0.0.0.0/0
          next-hop-address: 192.168.111.2
          next-hop-interface: eno1
          table-id: 254
EOF
```

- ❶ This IP address is used to determine which node performs the bootstrapping process as well as running the **assisted-service** component. You must provide the rendezvous IP address when you do not specify at least one host's IP address in the **networkConfig** parameter. If this address is not provided, one IP address is selected from the provided hosts' **networkConfig**.
- ❷ Optional: Host configuration. The number of hosts defined must not exceed the total number of hosts defined in the **install-config.yaml** file, which is the sum of the values of the **compute.replicas** and **controlPlane.replicas** parameters.
- ❸ Optional: Overrides the hostname obtained from either the Dynamic Host Configuration Protocol (DHCP) or a reverse DNS lookup. Each host must have a unique hostname supplied by one of these methods.
- ❹

Enables provisioning of the Red Hat Enterprise Linux CoreOS (RHCOS) image to a particular device. The installation program examines the devices in the order it discovers

- 5 Optional: Configures the network interface of a host in NMState format.

Additional resources

- [Configuring regions and zones for a VMware vCenter](#)
- [Verifying the supported architecture for installing an Agent-based installer cluster](#)
- [Configuring the Agent-based Installer to use mirrored images](#)

4.2.4. Creating additional manifest files

As an optional task, you can create additional manifests to further configure your cluster beyond the configurations available in the **install-config.yaml** and **agent-config.yaml** files.



IMPORTANT

Customizations to the cluster made by additional manifests are not validated, are not guaranteed to work, and might result in a nonfunctional cluster.

4.2.4.1. Creating a directory to contain additional manifests

If you create additional manifests to configure your Agent-based installation beyond the **install-config.yaml** and **agent-config.yaml** files, you must create an **openshift** subdirectory within your installation directory. All of your additional machine configurations must be located within this subdirectory.



NOTE

The most common type of additional manifest you can add is a **MachineConfig** object. For examples of **MachineConfig** objects you can add during the Agent-based installation, see "Using MachineConfig objects to configure nodes" in the "Additional resources" section.

Procedure

- On your installation host, create an **openshift** subdirectory within the installation directory by running the following command:

```
$ mkdir <installation_directory>/openshift
```

Additional resources

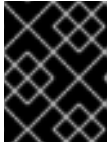
- [Using MachineConfig objects to configure nodes](#)

4.2.4.2. Disk partitioning

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

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

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



IMPORTANT

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

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

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

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

Prerequisites

- You have created an **openshift** subdirectory within your installation directory.

Procedure

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

```
variant: openshift
version: 4.18.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
    - device: /dev/disk/by-id/<device_name> ❶
  partitions:
    - label: var
      start_mib: <partition_start_offset> ❷
      size_mib: <partition_size> ❸
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
```

```
format: xfs
mount_options: [defaults, prjquota] 4
with_mount_unit: true
```

- 1 The storage device name of the disk that you want to partition.
- 2 When adding a data partition to the boot disk, a minimum offset value of 25000 mebibytes is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no offset value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.
- 3 The size of the data partition in mebibytes.
- 4 The **prjquota** mount option must be enabled for filesystems used for container storage.



NOTE

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

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

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

4.2.5. Using ZTP manifests

As an optional task, you can use GitOps Zero Touch Provisioning (ZTP) manifests to configure your installation beyond the options available through the **install-config.yaml** and **agent-config.yaml** files.



NOTE

GitOps ZTP manifests can be generated with or without configuring the **install-config.yaml** and **agent-config.yaml** files beforehand. If you chose to configure the **install-config.yaml** and **agent-config.yaml** files, the configurations will be imported to the ZTP cluster manifests when they are generated.

Prerequisites

- You have placed the **openshift-install** binary in a directory that is on your **PATH**.
- Optional: You have created and configured the **install-config.yaml** and **agent-config.yaml** files.

Procedure

1. Generate ZTP cluster manifests by running the following command:

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



IMPORTANT

If you have created the **install-config.yaml** and **agent-config.yaml** files, those files are deleted and replaced by the cluster manifests generated through this command.

Any configurations made to the **install-config.yaml** and **agent-config.yaml** files are imported to the ZTP cluster manifests when you run the **openshift-install agent create cluster-manifests** command.

2. Navigate to the **cluster-manifests** directory by running the following command:

```
$ cd <installation_directory>/cluster-manifests
```

3. Configure the manifest files in the **cluster-manifests** directory. For sample files, see the "Sample GitOps ZTP custom resources" section.
4. Disconnected clusters: If you did not define mirror configuration in the **install-config.yaml** file before generating the ZTP manifests, perform the following steps:
 - a. Navigate to the **mirror** directory by running the following command:

```
$ cd ../mirror
```

- b. Configure the manifest files in the **mirror** directory.

Additional resources

- [Sample GitOps ZTP custom resources](#) .
- See [Challenges of the network far edge](#) to learn more about GitOps Zero Touch Provisioning (ZTP).

4.2.6. Encrypting the disk

As an optional task, you can use this procedure to encrypt your disk or partition while installing OpenShift Container Platform with the Agent-based Installer.

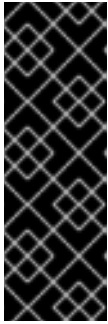
Prerequisites

- You have created and configured the **install-config.yaml** and **agent-config.yaml** files, unless you are using ZTP manifests.
- You have placed the **openshift-install** binary in a directory that is on your **PATH**.

Procedure

1. Generate ZTP cluster manifests by running the following command:

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



IMPORTANT

If you have created the **install-config.yaml** and **agent-config.yaml** files, those files are deleted and replaced by the cluster manifests generated through this command.

Any configurations made to the **install-config.yaml** and **agent-config.yaml** files are imported to the ZTP cluster manifests when you run the **openshift-install agent create cluster-manifests** command.



NOTE

If you have already generated ZTP manifests, skip this step.

2. Navigate to the **cluster-manifests** directory by running the following command:

```
$ cd <installation_directory>/cluster-manifests
```

3. Add the following section to the **agent-cluster-install.yaml** file:

```
diskEncryption:
  enableOn: all 1
  mode: tang 2
  tangServers: "server1": "http://tang-server-1.example.com:7500" 3
```

- 1** Specify which nodes to enable disk encryption on. Valid values are **none**, **all**, **masters**, and **workers**.
- 2** Specify which disk encryption mode to use. Valid values are **tpmv2** and **tang**.
- 3** Optional: If you are using Tang, specify the Tang servers.

Additional resources

- [About disk encryption](#)

4.2.7. Creating and booting the agent image

Use this procedure to boot the agent image on your machines.

Procedure

1. Create the agent image by running the following command:

```
$ openshift-install --dir <install_directory> agent create image
```



NOTE

Red Hat Enterprise Linux CoreOS (RHCOS) supports multipathing on the primary disk, allowing stronger resilience to hardware failure to achieve higher host availability. Multipathing is enabled by default in the agent ISO image, with a default **/etc/multipath.conf** configuration.

2. Boot the **agent.x86_64.iso**, **agent.aarch64.iso**, or **agent.s390x.iso** image on the bare metal machines.

4.2.8. Adding IBM Z agents with RHEL KVM

Use the following procedure to manually add IBM Z® agents with RHEL KVM. Only use this procedure for IBM Z® clusters with RHEL KVM.



NOTE

The **nmstateconfig** parameter must be configured for the KVM boot.

Procedure

1. Boot your RHEL KVM machine.
2. To deploy the virtual server, run the **virt-install** command with the following parameters:

ISO boot

```
$ virt-install
  --name <vm_name> \
  --autostart \
  --memory=<memory> \
  --cpu host \
  --vcpus=<vcpus> \
  --cdrom <path_to_image>/<agent_iso_image> \ 1
  --disk pool=default,size=<disk_pool_size> \
  --network network:default,mac=<mac_address> \
  --graphics none \
  --noautoconsole \
  --os-variant rhel9.0 \
  --wait=-1
```

- 1** For the **--cdrom** parameter, specify the location of the ISO image on the local server, for example, **<path_to_image>/home/<image>.iso**.

3. Optional: Enable FIPS mode.
To enable FIPS mode on IBM Z® clusters with RHEL KVM you must use PXE boot instead and run the **virt-install** command with the following parameters:

PXE boot

```
$ virt-install \
  --name <vm_name> \
  --autostart \
  --ram=16384 \
  --cpu host \
  --vcpus=8 \
  --location <path_to_kernel_initrd_image>,kernel=kernel.img,initrd=initrd.img \ 1
  --disk <qcow_image_path> \
  --network network:macvtap ,mac=<mac_address> \
  --graphics none \
```

```

--noautoconsole \
--wait=-1 \
--extra-args "rd.neednet=1 nameserver=<nameserver>" \
--extra-args "ip=<IP>::<nameserver>:<hostname>:enc1:none" \
--extra-args "coreos.live.rootfs_url=http://<http_server>:8080/agent.s390x-rootfs.img" \
--extra-args "random.trust_cpu=on rd.luks.options=discard" \
--extra-args "ignition.firstboot ignition.platform.id=metal" \
--extra-args "console=tty1 console=ttyS1,115200n8" \
--extra-args "coreos.inst.persistent-kargs=console=tty1 console=ttyS1,115200n8" \
--extra-args "fips=1" \ 2
--osinfo detect=on,require=off

```

- 1 For the **--location** parameter, specify the location of the kernel/initrd on the HTTP or HTTPS server.
- 2 To enable FIPS mode, specify **fips=1**. This entry is required in addition to setting the **fips** parameter to **true** in the **install-config.yaml** file.



NOTE

Currently, only PXE boot is supported to enable FIPS mode on IBM Z®.

4.2.9. Verifying that the current installation host can pull release images

After you boot the agent image and network services are made available to the host, the agent console application performs a pull check to verify that the current host can retrieve release images.

If the primary pull check passes, you can quit the application to continue with the installation. If the pull check fails, the application performs additional checks, as seen in the **Additional checks** section of the TUI, to help you troubleshoot the problem. A failure for any of the additional checks is not necessarily critical as long as the primary pull check succeeds.

If there are host network configuration issues that might cause an installation to fail, you can use the console application to make adjustments to your network configurations.



IMPORTANT

If the agent console application detects host network configuration issues, the installation workflow will be halted until the user manually stops the console application and signals the intention to proceed.

Procedure

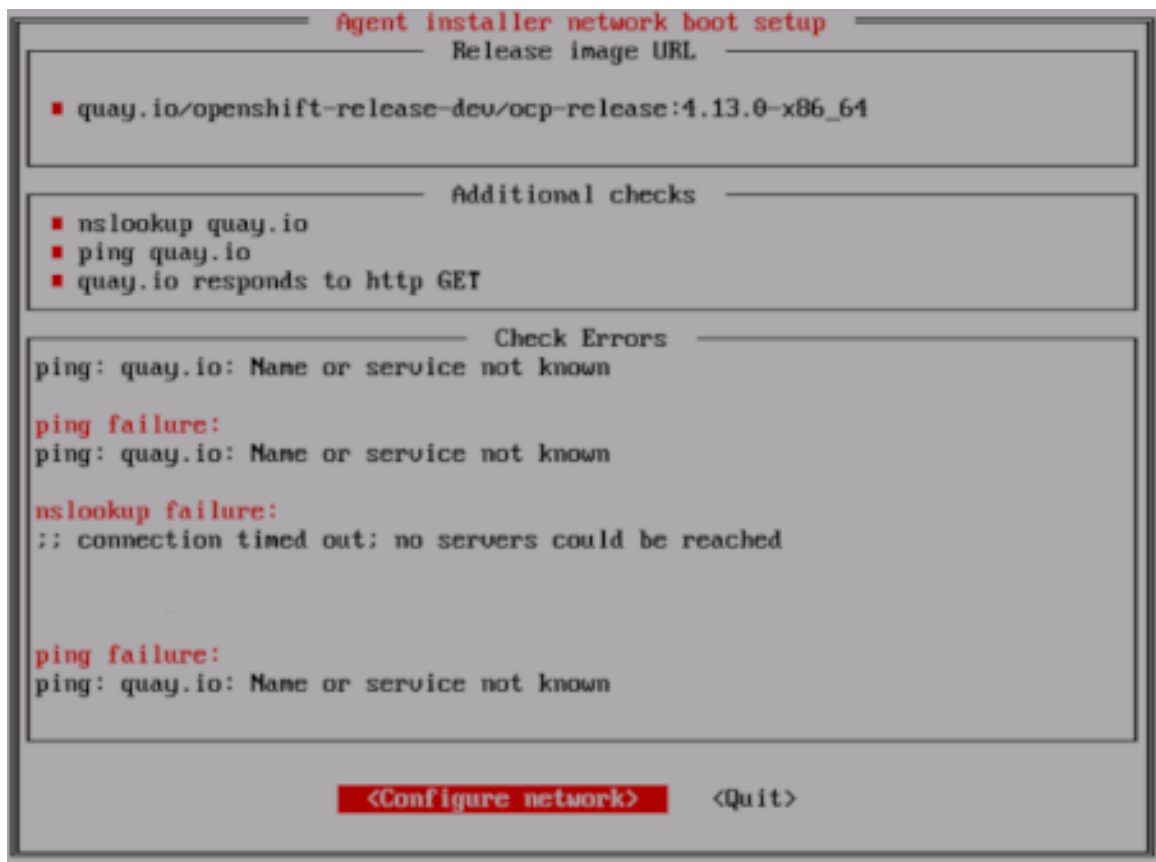
1. Wait for the agent console application to check whether or not the configured release image can be pulled from a registry.
2. If the agent console application states that the installer connectivity checks have passed, wait for the prompt to time out to continue with the installation.

**NOTE**

You can still choose to view or change network configuration settings even if the connectivity checks have passed.

However, if you choose to interact with the agent console application rather than letting it time out, you must manually quit the TUI to proceed with the installation.

3. If the agent console application checks have failed, which is indicated by a red icon beside the **Release image URL** pull check, use the following steps to reconfigure the host's network settings:
 - a. Read the **Check Errors** section of the TUI. This section displays error messages specific to the failed checks.



- b. Select **Configure network** to launch the NetworkManager TUI.
- c. Select **Edit a connection** and select the connection you want to reconfigure.
- d. Edit the configuration and select **OK** to save your changes.
- e. Select **Back** to return to the main screen of the NetworkManager TUI.
- f. Select **Activate a Connection**.
- g. Select the reconfigured network to deactivate it.
- h. Select the reconfigured network again to reactivate it.
- i. Select **Back** and then select **Quit** to return to the agent console application.

- j. Wait at least five seconds for the continuous network checks to restart using the new network configuration.
- k. If the **Release image URL** pull check succeeds and displays a green icon beside the URL, select **Quit** to exit the agent console application and continue with the installation.

4.2.10. Tracking and verifying installation progress

Use the following procedure to track installation progress and to verify a successful installation.

Prerequisites

- You have configured a DNS record for the Kubernetes API server.

Procedure

1. Optional: To know when the bootstrap host (rendezvous host) reboots, run the following command:

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

- 1 For **<install_directory>**, specify the path to the directory where the agent ISO was generated.
- 2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

Example output

```
.....
.....
INFO Bootstrap configMap status is complete
INFO cluster bootstrap is complete
```

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

2. To track the progress and verify successful installation, run the following command:

```
$ openshift-install --dir <install_directory> agent wait-for install-complete 1
```

- 1 For **<install_directory>** directory, specify the path to the directory where the agent ISO was generated.

Example output

```
.....
.....
INFO Cluster is installed
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run
```

INFO export KUBECONFIG=/home/core/installer/auth/kubeconfig
 INFO Access the OpenShift web-console here: <https://console-openshift-console.apps.sno-cluster.test.example.com>

NOTE

If you are using the optional method of GitOps ZTP manifests, you can configure IP address endpoints for cluster nodes through the **AgentClusterInstall.yaml** file in three ways:

- IPv4
- IPv6
- IPv4 and IPv6 in parallel (dual-stack)

IPv6 is supported only on bare metal platforms.

Example of dual-stack networking

```
apiVIP: 192.168.11.3
ingressVIP: 192.168.11.4
clusterDeploymentRef:
  name: mycluster
imageSetRef:
  name: openshift-4.18
networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
```

Additional resources

- See [Deploying with dual-stack networking](#).
- See [Configuring the install-config yaml file](#).
- See [Configuring a three-node cluster](#) to deploy three-node clusters in bare-metal environments.
- See [About root device hints](#).
- See [NMState state examples](#).

4.3. SAMPLE GITOPS ZTP CUSTOM RESOURCES

You can optionally use GitOps Zero Touch Provisioning (ZTP) custom resource (CR) objects to install an OpenShift Container Platform cluster with the Agent-based Installer.

You can customize the following GitOps ZTP custom resources to specify more details about your OpenShift Container Platform cluster. The following sample GitOps ZTP custom resources are for a single-node cluster.

Example `agent-cluster-install.yaml` file

```
apiVersion: extensions.hive.openshift.io/v1beta1
kind: AgentClusterInstall
metadata:
  name: test-agent-cluster-install
  namespace: cluster0
spec:
  clusterDeploymentRef:
    name: otest
  imageSetRef:
    name: openshift-4.18
  networking:
    clusterNetwork:
      - cidr: 10.128.0.0/14
        hostPrefix: 23
    serviceNetwork:
      - 172.30.0.0/16
  provisionRequirements:
    controlPlaneAgents: 1
    workerAgents: 0
  sshPublicKey: <ssh_public_key>
```

Example `cluster-deployment.yaml` file

```
apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: otest
  namespace: cluster0
spec:
  baseDomain: test.metalkube.org
  clusterInstallRef:
    group: extensions.hive.openshift.io
    kind: AgentClusterInstall
    name: test-agent-cluster-install
    version: v1beta1
  clusterName: otest
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    agentBareMetal:
      agentSelector:
        matchLabels:
          bla: aaa
  pullSecretRef:
    name: pull-secret
```

Example cluster-image-set.yaml file

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: openshift-4.18
spec:
  releaseImage: registry.ci.openshift.org/ocp/release:4.18.0-0.nightly-2022-06-06-025509
```

Example infra-env.yaml file

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: myinfraenv
  namespace: cluster0
spec:
  clusterRef:
    name: ostest
    namespace: cluster0
  cpuArchitecture: aarch64
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: <ssh_public_key>
  nmStateConfigLabelSelector:
    matchLabels:
      cluster0-nmstate-label-name: cluster0-nmstate-label-value
```

Example nmstateconfig.yaml file

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: master-0
  namespace: openshift-machine-api
  labels:
    cluster0-nmstate-label-name: cluster0-nmstate-label-value
spec:
  config:
    interfaces:
      - name: eth0
        type: ethernet
        state: up
        mac-address: 52:54:01:aa:aa:a1
        ipv4:
          enabled: true
          address:
            - ip: 192.168.122.2
              prefix-length: 23
          dhcp: false
    dns-resolver:
      config:
        server:
          - 192.168.122.1
```

```

routes:
  config:
    - destination: 0.0.0.0/0
      next-hop-address: 192.168.122.1
      next-hop-interface: eth0
      table-id: 254
interfaces:
  - name: "eth0"
    macAddress: 52:54:01:aa:aa:a1

```

Example pull-secret.yaml file

```

apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: pull-secret
  namespace: cluster0
stringData:
  .dockerconfigjson: <pull_secret>

```

Additional resources

- See [Challenges of the network far edge](#) to learn more about GitOps Zero Touch Provisioning (ZTP).

4.4. GATHERING LOG DATA FROM A FAILED AGENT-BASED INSTALLATION

Use the following procedure to gather log data about a failed Agent-based installation to provide for a support case.

Prerequisites

- You have configured a DNS record for the Kubernetes API server.

Procedure

1. Run the following command and collect the output:

```

$ ./openshift-install --dir <installation_directory> agent wait-for bootstrap-complete --log-level=debug

```

Example error message

```

...
ERROR Bootstrap failed to complete: : bootstrap process timed out: context deadline exceeded

```

2. If the output from the previous command indicates a failure, or if the bootstrap is not progressing, run the following command to connect to the rendezvous host and collect the output:

■

```
$ ssh core@<node-ip> agent-gather -O >agent-gather.tar.xz
```



NOTE

Red Hat Support can diagnose most issues using the data gathered from the rendezvous host, but if some hosts are not able to register, gathering this data from every host might be helpful.

3. If the bootstrap completes and the cluster nodes reboot, run the following command and collect the output:

```
$ ./openshift-install --dir <install_directory> agent wait-for install-complete --log-level=debug
```

4. If the output from the previous command indicates a failure, perform the following steps:
 - a. Export the **kubeconfig** file to your environment by running the following command:

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

- b. Gather information for debugging by running the following command:

```
$ oc adm must-gather
```

- c. Create a compressed file from the **must-gather** directory that was just created in your working directory by running the following command:

```
$ tar cvaf must-gather.tar.gz <must_gather_directory>
```

5. Excluding the **/auth** subdirectory, attach the installation directory used during the deployment to your support case on the [Red Hat Customer Portal](#).
6. Attach all other data gathered from this procedure to your support case.

CHAPTER 5. PREPARING PXE ASSETS FOR OPENSIFT CONTAINER PLATFORM

Use the following procedures to create the assets needed to PXE boot an OpenShift Container Platform cluster using the Agent-based Installer.

The assets you create in these procedures will deploy a single-node OpenShift Container Platform installation. You can use these procedures as a basis and modify configurations according to your requirements.

See [Installing an OpenShift Container Platform cluster with the Agent-based Installer](#) to learn about more configurations available with the Agent-based Installer.

5.1. PREREQUISITES

- You reviewed details about the [OpenShift Container Platform installation and update](#) processes.

5.2. DOWNLOADING THE AGENT-BASED INSTALLER

Use this procedure to download the Agent-based Installer and the CLI needed for your installation.

Procedure

1. Log in to the OpenShift Container Platform web console using your login credentials.
2. Navigate to [Datacenter](#).
3. Click **Run Agent-based Installer locally**.
4. Select the operating system and architecture for the **OpenShift Installer** and **Command line interface**.
5. Click **Download Installer** to download and extract the install program.
6. Download or copy the pull secret by clicking on **Download pull secret** or **Copy pull secret**
7. Click **Download command-line tools** and place the **openshift-install** binary in a directory that is on your **PATH**.

5.3. CREATING THE PREFERRED CONFIGURATION INPUTS

Use this procedure to create the preferred configuration inputs used to create the PXE files.



NOTE

Configuring the **install-config.yaml** and **agent-config.yaml** files is the preferred method for using the Agent-based Installer. Using GitOps ZTP manifests is optional.

Procedure

1. Install the **nmstate** dependency by running the following command:

```
$ sudo dnf install /usr/bin/nmstatectl -y
```

2. Place the **openshift-install** binary in a directory that is on your PATH.
3. Create a directory to store the install configuration by running the following command:

```
$ mkdir ~/<directory_name>
```

4. Create the **install-config.yaml** file by running the following command:

```
$ cat << EOF > ./<directory_name>/install-config.yaml
apiVersion: v1
baseDomain: test.example.com
compute:
- architecture: amd64 ❶
  hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  replicas: 1
metadata:
  name: sno-cluster ❷
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 192.168.0.0/16
  networkType: OVNKubernetes ❸
  serviceNetwork:
  - 172.30.0.0/16
platform: ❹
  none: {}
pullSecret: '<pull_secret>' ❺
sshKey: '<ssh_pub_key>' ❻
EOF
```

- ❶ Specify the system architecture. Valid values are **amd64**, **arm64**, **ppc64le**, and **s390x**.

If you are using the release image with the **multi** payload, you can install the cluster on different architectures such as **arm64**, **amd64**, **s390x**, and **ppc64le**. Otherwise, you can install the cluster only on the **release architecture** displayed in the output of the **openshift-install version** command. For more information, see "Verifying the supported architecture for installing an Agent-based Installer cluster".

- ❷ Required. Specify your cluster name.
- ❸ The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- ❹ Specify your platform.

**NOTE**

For bare-metal platforms, host settings made in the platform section of the **install-config.yaml** file are used by default, unless they are overridden by configurations made in the **agent-config.yaml** file.

- 5 Specify your pull secret.
- 6 Specify your SSH public key.

**NOTE**

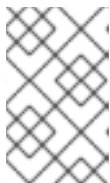
If you set the platform to **vSphere** or **baremetal**, you can configure IP address endpoints for cluster nodes in three ways:

- IPv4
- IPv6
- IPv4 and IPv6 in parallel (dual-stack)

IPv6 is supported only on bare metal platforms.

Example of dual-stack networking

```
networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
      - 192.168.11.3
      - 2001:DB8::4
    ingressVIPs:
      - 192.168.11.4
      - 2001:DB8::5
```

**NOTE**

When you use a disconnected mirror registry, you must add the certificate file that you created previously for your mirror registry to the **additionalTrustBundle** field of the **install-config.yaml** file.

5. Create the **agent-config.yaml** file by running the following command:

```
$ cat > agent-config.yaml << EOF
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 ❶
hosts: ❷
- hostname: master-0 ❸
  interfaces:
    - name: eno1
      macAddress: 00:ef:44:21:e6:a5
  rootDeviceHints: ❹
    deviceName: /dev/sdb
  networkConfig: ❺
    interfaces:
      - name: eno1
        type: ethernet
        state: up
        mac-address: 00:ef:44:21:e6:a5
      ipv4:
        enabled: true
        address:
          - ip: 192.168.111.80
            prefix-length: 23
        dhcp: false
    dns-resolver:
      config:
        server:
          - 192.168.111.1
    routes:
      config:
        - destination: 0.0.0.0/0
          next-hop-address: 192.168.111.2
          next-hop-interface: eno1
          table-id: 254
EOF
```

- ❶ This IP address is used to determine which node performs the bootstrapping process as well as running the **assisted-service** component. You must provide the rendezvous IP address when you do not specify at least one host's IP address in the **networkConfig** parameter. If this address is not provided, one IP address is selected from the provided hosts' **networkConfig**.
- ❷ Optional: Host configuration. The number of hosts defined must not exceed the total number of hosts defined in the **install-config.yaml** file, which is the sum of the values of the **compute.replicas** and **controlPlane.replicas** parameters.
- ❸ Optional: Overrides the hostname obtained from either the Dynamic Host Configuration Protocol (DHCP) or a reverse DNS lookup. Each host must have a unique hostname supplied by one of these methods.
- ❹

Enables provisioning of the Red Hat Enterprise Linux CoreOS (RHCOS) image to a particular device. The installation program examines the devices in the order it discovers

- 5 Optional: Configures the network interface of a host in NMState format.

6. Optional: To create an iPXE script, add the **bootArtifactsBaseURL** to the **agent-config.yaml** file:

```
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80
bootArtifactsBaseURL: <asset_server_URL>
```

Where **<asset_server_URL>** is the URL of the server you will upload the PXE assets to.

Additional resources

- [Deploying with dual-stack networking](#).
- [Configuring the install-config yaml file](#).
- See [Configuring a three-node cluster](#) to deploy three-node clusters in bare metal environments.
- [About root device hints](#).
- [NMState state examples](#).
- [Optional: Creating additional manifest files](#)

5.4. CREATING THE PXE ASSETS

Use the following procedure to create the assets and optional script to implement in your PXE infrastructure.

Procedure

1. Create the PXE assets by running the following command:

```
$ openshift-install agent create pxe-files
```

The generated PXE assets and optional iPXE script can be found in the **boot-artifacts** directory.

Example filesystem with PXE assets and optional iPXE script

```
boot-artifacts
├── agent.x86_64-initrd.img
├── agent.x86_64.ipxe
├── agent.x86_64-rootfs.img
└── agent.x86_64-vmlinuz
```

**IMPORTANT**

The contents of the **boot-artifacts** directory vary depending on the specified architecture.

**NOTE**

Red Hat Enterprise Linux CoreOS (RHCOS) supports multipathing on the primary disk, allowing stronger resilience to hardware failure to achieve higher host availability. Multipathing is enabled by default in the agent ISO image, with a default **/etc/multipath.conf** configuration.

2. Upload the PXE assets and optional script to your infrastructure where they will be accessible during the boot process.

**NOTE**

If you generated an iPXE script, the location of the assets must match the **bootArtifactsBaseURL** you added to the **agent-config.yaml** file.

5.5. MANUALLY ADDING IBM Z AGENTS

After creating the PXE assets, you can add IBM Z® agents. Only use this procedure for IBM Z® clusters.

Depending on your IBM Z® environment, you can choose from the following options:

- Adding IBM Z® agents with z/VM
- Adding IBM Z® agents with RHEL KVM
- Adding IBM Z® agents with Logical Partition (LPAR)

**NOTE**

Currently, ISO boot support on IBM Z® (**s390x**) is available only for Red Hat Enterprise Linux (RHEL) KVM, which provides the flexibility to choose either PXE or ISO-based installation. For installations with z/VM and Logical Partition (LPAR), only PXE boot is supported.

5.5.1. Networking requirements for IBM Z

In IBM Z environments, advanced networking technologies such as Open Systems Adapter (OSA), HiperSockets, and Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) require specific configurations that deviate from the standard network settings and those needs to be persisted for multiple boot scenarios that occur in the Agent-based Installation.

To persist these parameters during boot, the **ai.ip_cfg_override=1** parameter is required in the **.parm** file. This parameter is used with the configured network cards to ensure a successful and efficient deployment on IBM Z.

The following table lists the network devices that are supported on each hypervisor for the network configuration override functionality:

Network device	z/VM	KVM	LPAR Classic	LPAR Dynamic Partition Manager (DPM)
Virtual Switch	Supported ^[1]	Not applicable ^[2]	Not applicable	Not applicable
Direct attached Open Systems Adapter (OSA)	Supported	Not required ^[3]	Supported	Not required
RDMA over Converged Ethernet (RoCE)	Not required	Not required	Not required	Not required
HiperSockets	Supported	Not required	Supported	Not required

1. Supported: When the **ai.ip_cfg_override** parameter is required for the installation procedure.
2. Not Applicable: When a network card is not applicable to be used on the hypervisor.
3. Not required: When the **ai.ip_cfg_override** parameter is not required for the installation procedure.

5.5.2. Configuring network overrides in an IBM Z environment

You can specify a static IP address on IBM Z machines that use Logical Partition (LPAR) and z/VM. This is useful when the network devices do not have a static MAC address assigned to them.



NOTE

If you are using an OSA network device in Processor Resource/Systems Manager (PR/SM) mode, the lack of persistent MAC addresses can lead to a dynamic assignment of roles for nodes. This means that the roles of individual nodes are not fixed and can change, as the system is unable to reliably associate specific MAC addresses with designated node roles. If MAC addresses are not persistent for any of the interfaces, roles for the nodes are assigned randomly during Agent-based installation.

Procedure

- If you have an existing **.parm** file, edit it to include the following entry:

```
ai.ip_cfg_override=1
```

This parameter allows the file to add the network settings to the Red Hat Enterprise Linux CoreOS (RHCOS) installer.

Example .parm file

```
rd.neednet=1 cio_ignore=all,!condev
console=ttysclp0
coreos.live.rootfs_url=<coreos_url> 1
ip=<ip>::<gateway>:<netmask>:<hostname>::none nameserver=<dns>
```

```
rd.znet=qeth,<network_adaptor_range>,layer2=1
rd.<disk_type>=<adapter> ❷
rd.zfcp=<adapter>,<wwpn>,<lun> random.trust_cpu=on ❸
zfcp.allow_lun_scan=0
ai.ip_cfg_override=1
ignition.firstboot ignition.platform.id=metal
random.trust_cpu=on
```

- ❶ For the **coreos.live.rootfs_url** artifact, specify the matching **rootfs** artifact for the **kernel** and **initramfs** that you are booting. Only HTTP and HTTPS protocols are supported.
- ❷ For installations on direct access storage devices (DASD) type disks, use **rd.** to specify the DASD where Red Hat Enterprise Linux CoreOS (RHCOS) is to be installed. For installations on Fibre Channel Protocol (FCP) disks, use **rd.zfcp=<adapter>,<wwpn>,<lun>** to specify the FCP disk where RHCOS is to be installed.
- ❸ Specify values for **adapter**, **wwpn**, and **lun** as in the following example:
rd.zfcp=0.0.8002,0x500507630400d1e3,0x4000404600000000.



NOTE

The **override** parameter overrides the host's network configuration settings.

5.5.3. Adding IBM Z agents with z/VM

Use the following procedure to manually add IBM Z® agents with z/VM. Only use this procedure for IBM Z® clusters with z/VM.

Prerequisites

- A running file server with access to the guest Virtual Machines.

Procedure

1. Create a parameter file for the z/VM guest:

Example parameter file

```
rd.neednet=1 \
console=ttyS0 \
coreos.live.rootfs_url=<rootfs_url> \ ❶
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1 \ ❷
zfcp.allow_lun_scan=0 \ ❸
ai.ip_cfg_override=1 \
rd.znet=qeth,0.0.bdd0,0.0.bdd1,0.0.bdd2,layer2=1 \
rd.dasd=0.0.4411 \ ❹
rd.zfcp=0.0.8001,0x50050763040051e3,0x4000406300000000 \ ❺
fips=1 \ ❻
random.trust_cpu=on rd.luks.options=discard \
ignition.firstboot ignition.platform.id=metal \
console=tty1 console=ttyS1,115200n8 \
coreos.inst.persistent-kargs="console=tty1 console=ttyS1,115200n8"
```

- 1 For the **coreos.live.rootfs_url** artifact, specify the matching **rootfs** artifact for the **kernel** and **initramfs** that you are booting. Only HTTP and HTTPS protocols are supported.
- 2 For the **ip** parameter, assign the IP address automatically using DHCP, or manually assign the IP address, as described in "Installing a cluster with z/VM on IBM Z® and IBM® LinuxONE".
- 3 The default is **1**. Omit this entry when using an OSA network adapter.
- 4 For installations on DASD-type disks, use **rd.dasd** to specify the DASD where Red Hat Enterprise Linux CoreOS (RHCOS) is to be installed. Omit this entry for FCP-type disks.
- 5 For installations on FCP-type disks, use **rd.zfcp=<adapter>,<wwpn>,<lun>** to specify the FCP disk where RHCOS is to be installed. Omit this entry for DASD-type disks.
- 6 To enable FIPS mode, specify **fips=1**. This entry is required in addition to setting the **fips** parameter to **true** in the **install-config.yaml** file.

Leave all other parameters unchanged.

2. Punch the **kernel.img**, **generic.parm**, and **initrd.img** files to the virtual reader of the z/VM guest virtual machine.
For more information, see [PUNCH](#) (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.

3. Log in to the conversational monitor system (CMS) on the bootstrap machine.
4. IPL the bootstrap machine from the reader by running the following command:

```
$ ipl c
```

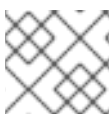
For more information, see [IPL](#) (IBM Documentation).

Additional resources

- [Installing a cluster with z/VM on IBM Z and IBM LinuxONE](#)

5.5.4. Adding IBM Z agents with RHEL KVM

Use the following procedure to manually add IBM Z® agents with RHEL KVM. Only use this procedure for IBM Z® clusters with RHEL KVM.



NOTE

The **nmstateconfig** parameter must be configured for the KVM boot.

Procedure

1. Boot your RHEL KVM machine.

- To deploy the virtual server, run the **virt-install** command with the following parameters:

```
$ virt-install \
  --name <vm_name> \
  --autostart \
  --ram=16384 \
  --cpu host \
  --vcpus=8 \
  --location <path_to_kernel_initrd_image>,kernel=kernel.img,initrd=initrd.img 1 \
  --disk <qcow_image_path> \
  --network network:macvtap ,mac=<mac_address> \
  --graphics none \
  --noautoconsole \
  --wait=-1 \
  --extra-args "rd.neednet=1 nameserver=<nameserver>" \
  --extra-args "ip=<IP>::<nameserver>::<hostname>:enc1:none" \
  --extra-args "coreos.live.rootfs_url=http://<http_server>:8080/agent.s390x-rootfs.img" \
  --extra-args "random.trust_cpu=on rd.luks.options=discard" \
  --extra-args "ignition.firstboot ignition.platform.id=metal" \
  --extra-args "console=tty1 console=ttyS1,115200n8" \
  --extra-args "coreos.inst.persistent-kargs=console=tty1 console=ttyS1,115200n8" \
  --osinfo detect=on,require=off
```

- 1** For the **--location** parameter, specify the location of the **kernel** and **initrd** files. The location can be a local server path or a URL using HTTP or HTTPS.

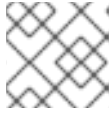
- Optional: Enable FIPS mode.

To enable FIPS mode on IBM Z® clusters with RHEL KVM you must use PXE boot instead and run the **virt-install** command with the following parameters:

PXE boot

```
$ virt-install \
  --name <vm_name> \
  --autostart \
  --ram=16384 \
  --cpu host \
  --vcpus=8 \
  --location <path_to_kernel_initrd_image>,kernel=kernel.img,initrd=initrd.img 1 \
  --disk <qcow_image_path> \
  --network network:macvtap ,mac=<mac_address> \
  --graphics none \
  --noautoconsole \
  --wait=-1 \
  --extra-args "rd.neednet=1 nameserver=<nameserver>" \
  --extra-args "ip=<IP>::<nameserver>::<hostname>:enc1:none" \
  --extra-args "coreos.live.rootfs_url=http://<http_server>:8080/agent.s390x-rootfs.img" \
  --extra-args "random.trust_cpu=on rd.luks.options=discard" \
  --extra-args "ignition.firstboot ignition.platform.id=metal" \
  --extra-args "console=tty1 console=ttyS1,115200n8" \
  --extra-args "coreos.inst.persistent-kargs=console=tty1 console=ttyS1,115200n8" \
  --extra-args "fips=1" 2 \
  --osinfo detect=on,require=off
```


- 1 For the **--location** parameter, specify the location of the kernel/initrd on the HTTP or HTTPS server.
- 2 To enable FIPS mode, specify **fips=1**. This entry is required in addition to setting the **fips** parameter to **true** in the **install-config.yaml** file.



NOTE

Currently, only PXE boot is supported to enable FIPS mode on IBM Z®.

Additional resources

- [Installing a cluster with RHEL KVM on IBM Z and IBM LinuxONE](#)

5.5.5. Adding IBM Z agents in a Logical Partition (LPAR)

Use the following procedure to manually add IBM Z® agents to your cluster that runs in an LPAR environment. Use this procedure only for IBM Z® clusters running in an LPAR.

Prerequisites

- You have Python 3 installed.
- A running file server with access to the Logical Partition (LPAR).

Procedure

1. Create a boot parameter file for the agents.

Example parameter file

```
rd.neednet=1 cio_ignore=all,!condev \
console=ttySCLP0 \
ignition.firstboot ignition.platform.id=metal
coreos.live.rootfs_url=http://<http_server>/rhcos-<version>-live-rootfs.<architecture>.img \ 1
coreos.inst.persistent-kargs=console=ttySCLP0 \
ip=<ip>::<gateway>:<netmask>:<hostname>::none nameserver=<dns> \ 2
rd.znet=qeth,<network_adaptor_range>,layer2=1
rd.<disk_type>=<adaptor> \ 3
fips=1 \ 4
zfcp.allow_lun_scan=0 \
ai.ip_cfg_override=1 \
random.trust_cpu=on rd.luks.options=discard
```

- 1 For the **coreos.live.rootfs_url** artifact, specify the matching **rootfs** artifact for the **kernel** and **initramfs** that you are starting. Only HTTP and HTTPS protocols are supported.
- 2 For the **ip** parameter, manually assign the IP address, as described in *Installing a cluster with z/VM on IBM Z and IBM LinuxONE*.
- 3 For installations on DASD-type disks, use **rd.dasd** to specify the DASD where Red Hat Enterprise Linux CoreOS (RHCOS) is to be installed. For installations on FCP-type disks, use **rd.zfcp=<adaptor>,<wwpn>,<lun>** to specify the FCP disk where RHCOS is to be

installed.

- 4 To enable FIPS mode, specify **fips=1**. This entry is required in addition to setting the **fips** parameter to **true** in the **install-config.yaml** file.



NOTE

The **.ins** and **initrd.img.addrsz** files are automatically generated for **s390x** architecture as part of boot-artifacts from the installation program and are only used when booting in an LPAR environment.

Example filesystem with LPAR boot

```
boot-artifacts
├── agent.s390x-generic.ins
├── agent.s390x-initrd.addrsz
├── agent.s390x-rootfs.img
├── agent.s390x-kernel.img
└── agent.s390x-rootfs.img
```

2. Rename the **boot-artifacts** file present in the **generic.ins** parameter file to match the names of the **boot-artifacts** file generated by the installation program.
3. Transfer the **initrd**, **kernel**, **generic.ins**, and **initrd.img.addrsz** parameter files to the file server. For more information, see [Booting Linux in LPAR mode](#) (IBM documentation).
4. Start the machine.
5. Repeat the procedure for all other machines in the cluster.

Additional resources

- [Installing a cluster in an LPAR on IBM Z and IBM LinuxONE](#)

5.5.6. Adding IBM Z agents in a Logical Partition (LPAR) to an existing cluster

In OpenShift Container Platform 4.18, the **.ins** and **initrd.img.addrsz** files are not automatically generated as part of the boot-artifacts for an existing cluster. You must manually generate these files for IBM Z® clusters running in an LPAR.

The **.ins** file is a special file that includes installation data and is present on the FTP server. You can access this file from the hardware management console (HMC) system. This file contains details such as mapping of the location of installation data on the disk or FTP server and the memory locations where the data is to be copied.

Prerequisites

- A running file server with access to the LPAR.

Procedure

1. Generate the **.ins** and **initrd.img.addrsz** files:

- a. Retrieve the size of the **kernel** and **initrd** by running the following commands:

```
$ KERNEL_IMG_PATH='./kernel.img'

$ INITRD_IMG_PATH='./initrd.img'

$ CMDLINE_PATH='./generic.prm'

$ kernel_size=$(stat -c%s $KERNEL_IMG_PATH)

$ initrd_size=$(stat -c%s $INITRD_IMG_PATH)
```

- b. Round up the **kernel** size to the next Mebibytes (MiB) boundary by running the following command:

```
$ BYTE_PER_MIB=$(( 1024 * 1024 )) offset=$(( (kernel_size + BYTE_PER_MIB - 1) /
BYTE_PER_MIB * BYTE_PER_MIB ))
```

- c. Create the kernel binary patch file that contains the **initrd** address and size by running the following commands:

```
$ INITRD_IMG_NAME=$(echo $INITRD_IMG_PATH | rev | cut -d '/' -f 1 | rev)

$ KERNEL_OFFSET=0x00000000

$ KERNEL_CMDLINE_OFFSET=0x00010480

$ INITRD_ADDR_SIZE_OFFSET=0x00010408

$ OFFSET_HEX=$(printf '0x%08x\n' offset)
```

- d. Convert the address and size to binary format by running the following command:

```
$ printf "$(printf '%016x\n' $initrd_size)" | xxd -r -p > temp_size.bin
```

- e. Merge the address and size binaries by running the following command:

```
$ cat temp_address.bin temp_size.bin > "$INITRD_IMG_NAME.addrsize"
```

- f. Clean up temporary files by running the following command:

```
$ rm -rf temp_address.bin temp_size.bin
```

- g. Create the **.ins** file by running the following command:

```
$ cat > generic.ins <<EOF
$KERNEL_IMG_PATH $KERNEL_OFFSET
$INITRD_IMG_PATH $OFFSET_HEX
```

```
$INITRD_IMG_NAME.addrsize $INITRD_ADDR_SIZE_OFFSET  
$CMDLINE_PATH $KERNEL_CMDLINE_OFFSET  
EOF
```



NOTE

The file is based on the paths of the **kernel.img**, **initrd.img**, **initrd.img.addrsize**, and **cmdline** files and the memory locations where the data is to be copied.

2. Transfer the **initrd**, **kernel**, **generic.ins**, and **initrd.img.addrsize** parameter files to the file server. For more information, see [Booting Linux in LPAR mode](#) (IBM documentation).
3. Start the machine.
4. Repeat the procedure for all other machines in the cluster.

CHAPTER 6. PREPARING INSTALLATION ASSETS FOR ISCSI BOOTING

You can boot an OpenShift Container Platform cluster through Internet Small Computer System Interface (iSCSI) by using an ISO image generated by the Agent-based Installer. The following procedures describe how to prepare the necessary installation resources to boot from an iSCSI target.

The assets you create in these procedures deploy a single-node OpenShift Container Platform installation. You can use these procedures as a basis and modify configurations according to your requirements.

6.1. REQUIREMENTS FOR ISCSI BOOTING

The following configurations are necessary to enable iSCSI booting when using the Agent-based Installer:

- Dynamic Host Configuration Protocol (DHCP) must be configured. Static networking is not supported.
- You must create an additional network for iSCSI that is separate from the machine network of the cluster. The machine network is rebooted during cluster installation and cannot be used for the iSCSI session.
- If the host on which you are booting the agent ISO image also has an installed disk, it might be necessary to specify the iSCSI disk name in the **rootDeviceHints** parameter to ensure that it is chosen as the boot disk for the final Red Hat Enterprise Linux CoreOS (RHCOS) image. You can also use a diskless environment for iSCSI booting, in which case you do not need to set the **rootDeviceHints** parameter.

Additional resources

- [DHCP](#)
- [About root device hints](#)

6.2. PREREQUISITES

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

6.3. DOWNLOADING THE AGENT-BASED INSTALLER

Use this procedure to download the Agent-based Installer and the CLI needed for your installation.

Procedure

1. Log in to the OpenShift Container Platform web console using your login credentials.

2. Navigate to [Datacenter](#).
3. Click **Run Agent-based Installer locally**.
4. Select the operating system and architecture for the **OpenShift Installer** and **Command line interface**.
5. Click **Download Installer** to download and extract the install program.
6. Download or copy the pull secret by clicking on **Download pull secret** or **Copy pull secret**.
7. Click **Download command-line tools** and place the **openshift-install** binary in a directory that is on your **PATH**.

6.4. CREATING THE PREFERRED CONFIGURATION INPUTS

Use this procedure to create the preferred configuration inputs used to create the agent image.



NOTE

Configuring the **install-config.yaml** and **agent-config.yaml** files is the preferred method for using the Agent-based Installer. Using GitOps ZTP manifests is optional.

Procedure

1. Install the **nmstate** dependency by running the following command:

```
$ sudo dnf install /usr/bin/nmstatectl -y
```

2. Place the **openshift-install** binary in a directory that is on your PATH.
3. Create a directory to store the install configuration by running the following command:

```
$ mkdir ~/<directory_name>
```

4. Create the **install-config.yaml** file by running the following command:

```
$ cat << EOF > ./<directory_name>/install-config.yaml
apiVersion: v1
baseDomain: test.example.com
compute:
- architecture: amd64 1
  hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  replicas: 1
metadata:
  name: sno-cluster 2
networking:
  clusterNetwork:
```

```

- cidr: 10.128.0.0/14
  hostPrefix: 23
  machineNetwork:
- cidr: 192.168.0.0/16
  networkType: OVNKubernetes 3
  serviceNetwork:
- 172.30.0.0/16
platform: 4
  none: {}
pullSecret: '<pull_secret>' 5
sshKey: '<ssh_pub_key>' 6
EOF

```

- 1** Specify the system architecture. Valid values are **amd64**, **arm64**, **ppc64le**, and **s390x**.

If you are using the release image with the **multi** payload, you can install the cluster on different architectures such as **arm64**, **amd64**, **s390x**, and **ppc64le**. Otherwise, you can install the cluster only on the **release architecture** displayed in the output of the **openshift-install version** command. For more information, see "Verifying the supported architecture for installing an Agent-based Installer cluster".

- 2** Required. Specify your cluster name.
- 3** The cluster network plugin to install. The default value **OVNKubernetes** is the only supported value.
- 4** Specify your platform.



NOTE

For bare-metal platforms, host settings made in the platform section of the **install-config.yaml** file are used by default, unless they are overridden by configurations made in the **agent-config.yaml** file.

- 5** Specify your pull secret.
- 6** Specify your SSH public key.



NOTE

If you set the platform to **vSphere** or **baremetal**, you can configure IP address endpoints for cluster nodes in three ways:

- IPv4
- IPv6
- IPv4 and IPv6 in parallel (dual-stack)

IPv6 is supported only on bare metal platforms.

Example of dual-stack networking

```

networking:
  clusterNetwork:
    - cidr: 172.21.0.0/16
      hostPrefix: 23
    - cidr: fd02::/48
      hostPrefix: 64
  machineNetwork:
    - cidr: 192.168.11.0/16
    - cidr: 2001:DB8::/32
  serviceNetwork:
    - 172.22.0.0/16
    - fd03::/112
  networkType: OVNKubernetes
platform:
  baremetal:
    apiVIPs:
      - 192.168.11.3
      - 2001:DB8::4
    ingressVIPs:
      - 192.168.11.4
      - 2001:DB8::5

```

**NOTE**

When you use a disconnected mirror registry, you must add the certificate file that you created previously for your mirror registry to the **additionalTrustBundle** field of the **install-config.yaml** file.

5. Create the **agent-config.yaml** file by running the following command:

```

$ cat > agent-config.yaml << EOF
apiVersion: v1beta1
kind: AgentConfig
metadata:
  name: sno-cluster
rendezvousIP: 192.168.111.80 ❶
hosts: ❷
  - hostname: master-0 ❸
    interfaces:
      - name: eno1
        macAddress: 00:ef:44:21:e6:a5
    rootDeviceHints: ❹
      deviceName: /dev/sdb
    networkConfig: ❺
      interfaces:
        - name: eno1
          type: ethernet
          state: up
          mac-address: 00:ef:44:21:e6:a5
          ipv4:
            enabled: true
            address:
              - ip: 192.168.111.80
                prefix-length: 23

```



```

    dhcp: false
  dns-resolver:
    config:
      server:
        - 192.168.111.1
  routes:
    config:
      - destination: 0.0.0.0/0
        next-hop-address: 192.168.111.2
        next-hop-interface: eno1
        table-id: 254
  minimalISO: true 6
EOF

```

- 1** This IP address is used to determine which node performs the bootstrapping process as well as running the **assisted-service** component. You must provide the rendezvous IP address when you do not specify at least one host's IP address in the **networkConfig** parameter. If this address is not provided, one IP address is selected from the provided hosts' **networkConfig**.
- 2** Optional: Host configuration. The number of hosts defined must not exceed the total number of hosts defined in the **install-config.yaml** file, which is the sum of the values of the **compute.replicas** and **controlPlane.replicas** parameters.
- 3** Optional: Overrides the hostname obtained from either the Dynamic Host Configuration Protocol (DHCP) or a reverse DNS lookup. Each host must have a unique hostname supplied by one of these methods.
- 4** Enables provisioning of the Red Hat Enterprise Linux CoreOS (RHCOS) image to a particular device. The installation program examines the devices in the order it discovers them, and compares the discovered values with the hint values. It uses the first discovered device that matches the hint value.
- 5** Optional: Configures the network interface of a host in NMState format.
- 6** Generates an ISO image without the rootfs image file, and instead provides details about where to pull the rootfs file from. You must set this parameter to **true** to enable iSCSI booting.

Additional resources

- [Deploying with dual-stack networking](#)
- [Configuring the install-config.yaml file](#)
- [Configuring a three-node cluster](#)
- [About root device hints](#)
- [NMState state examples](#) (NMState documentation)
- [Optional: Creating additional manifest files](#)
- [Verifying the supported architecture for an Agent-based installation](#)

6.5. CREATING THE INSTALLATION FILES

Use the following procedure to generate the ISO image and create an iPXE script to upload to your iSCSI target.

Procedure

1. Create the agent image by running the following command:

```
$ openshift-install --dir <install_directory> agent create image
```

2. Create an iPXE script by running the following command:

```
$ cat << EOF > agent.ipxe
!ipxe
set initiator-iqn <iscsi_initiator_base>:\${hostname}
sanboot --keep iscsi:<iscsi_network_subnet>.1:::<iscsi_target_base>:\${hostname}
EOF
```

where:

<iscsi_initiator_base>

Specifies the iSCSI initiator name on the host that is booting the ISO. This name can also be used by the iSCSI target.

<iscsi_network_subnet>

Specifies the IP address of the iSCSI target.

<iscsi_target_base>

Specifies the iSCSI target name. This name can be the same as the initiator name.

Example Command

```
$ cat << EOF > agent.ipxe
!ipxe
set initiator-iqn iqn.2023-01.com.example:\${hostname}
sanboot --keep iscsi:192.168.45.1:::iqn.2023-01.com.example:\${hostname}
EOF
```

CHAPTER 7. PREPARING AN AGENT-BASED INSTALLED CLUSTER FOR THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR

You can install the multicluster engine Operator and deploy a hub cluster with the Agent-based OpenShift Container Platform Installer. The following procedure is partially automated and requires manual steps after the initial cluster is deployed.

7.1. PREREQUISITES

- You have read the following documentation:
 - [Cluster lifecycle with multicluster engine operator overview](#) .
 - [Persistent storage using local volumes](#) .
 - [Using GitOps ZTP to provision clusters at the network far edge](#) .
 - [Preparing to install with the Agent-based Installer](#) .
 - [About disconnected installation mirroring](#) .
- You have access to the internet to obtain the necessary container images.
- You have installed the OpenShift CLI (**oc**).
- If you are installing in a disconnected environment, you must have a configured local mirror registry for disconnected installation mirroring.

7.2. PREPARING AN AGENT-BASED CLUSTER DEPLOYMENT FOR THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR WHILE DISCONNECTED

You can mirror the required OpenShift Container Platform container images, the multicluster engine Operator, and the Local Storage Operator (LSO) into your local mirror registry in a disconnected environment. Ensure that you note the local DNS hostname and port of your mirror registry.



NOTE

To mirror your OpenShift Container Platform image repository to your mirror registry, you can use either the **oc adm release image** or **oc mirror** command. In this procedure, the **oc mirror** command is used as an example.

Procedure

1. Create an **<assets_directory>** folder to contain valid **install-config.yaml** and **agent-config.yaml** files. This directory is used to store all the assets.
2. To mirror an OpenShift Container Platform image repository, the multicluster engine, and the LSO, create a **ImageSetConfiguration.yaml** file with the following settings:

Example ImageSetConfiguration.yaml

```

kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
archiveSize: 4 1
storageConfig: 2
  imageURL: <your-local-registry-dns-name>:<your-local-registry-port>/mirror/oc-mirror-
  metadata 3
  skipTLS: true
mirror:
  platform:
    architectures:
      - "amd64"
    channels:
      - name: stable-4.18 4
      type: ocp
  additionalImages:
    - name: registry.redhat.io/ubi9/ubi:latest
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18 5
    packages: 6
      - name: multicluster-engine 7
      - name: local-storage-operator 8

```

- ¹ Specify the maximum size, in GiB, of each file within the image set.
- ² Set the back-end location to receive the image set metadata. This location can be a registry or local directory. It is required to specify **storageConfig** values.
- ³ Set the registry URL for the storage backend.
- ⁴ Set the channel that contains the OpenShift Container Platform images for the version you are installing.
- ⁵ Set the Operator catalog that contains the OpenShift Container Platform images that you are installing.
- ⁶ Specify only certain Operator packages and channels to include in the image set. Remove this field to retrieve all packages in the catalog.
- ⁷ The multicluster engine packages and channels.
- ⁸ The LSO packages and channels.



NOTE

This file is required by the **oc mirror** command when mirroring content.

3. To mirror a specific OpenShift Container Platform image repository, the multicluster engine, and the LSO, run the following command:

```
$ oc mirror --dest-skip-tls --config ocp-mce-imageset.yaml docker://<your-local-registry-dns-name>:<your-local-registry-port>
```

4. Update the registry and certificate in the **install-config.yaml** file:

Example imageContentSources.yaml

```
imageContentSources:
- source: "quay.io/openshift-release-dev/ocp-release"
  mirrors:
    - "<your-local-registry-dns-name>:<your-local-registry-port>/openshift/release-images"
- source: "quay.io/openshift-release-dev/ocp-v4.0-art-dev"
  mirrors:
    - "<your-local-registry-dns-name>:<your-local-registry-port>/openshift/release"
- source: "registry.redhat.io/ubi9"
  mirrors:
    - "<your-local-registry-dns-name>:<your-local-registry-port>/ubi9"
- source: "registry.redhat.io/multicluster-engine"
  mirrors:
    - "<your-local-registry-dns-name>:<your-local-registry-port>/multicluster-engine"
- source: "registry.redhat.io/rhel8"
  mirrors:
    - "<your-local-registry-dns-name>:<your-local-registry-port>/rhel8"
- source: "registry.redhat.io/redhat"
  mirrors:
    - "<your-local-registry-dns-name>:<your-local-registry-port>/redhat"
```

Additionally, ensure your certificate is present in the **additionalTrustBundle** field of the **install-config.yaml**.

Example install-config.yaml

```
additionalTrustBundle: |
-----BEGIN CERTIFICATE-----
ZZZZZZZZZZZZ
-----END CERTIFICATE-----
```



IMPORTANT

The **oc mirror** command creates a folder called **oc-mirror-workspace** with several outputs. This includes the **imageContentSourcePolicy.yaml** file that identifies all the mirrors you need for OpenShift Container Platform and your selected Operators.

5. Generate the cluster manifests by running the following command:

```
$ openshift-install agent create cluster-manifests
```

This command updates the cluster manifests folder to include a **mirror** folder that contains your mirror configuration.

7.3. PREPARING AN AGENT-BASED CLUSTER DEPLOYMENT FOR THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR WHILE CONNECTED

Create the required manifests for the multicluster engine Operator, the Local Storage Operator (LSO), and to deploy an agent-based OpenShift Container Platform cluster as a hub cluster.

Procedure

1. Create a sub-folder named **openshift** in the **<assets_directory>** folder. This sub-folder is used to store the extra manifests that will be applied during the installation to further customize the deployed cluster. The **<assets_directory>** folder contains all the assets including the **install-config.yaml** and **agent-config.yaml** files.



NOTE

The installer does not validate extra manifests.

2. For the multicluster engine, create the following manifests and save them in the **<assets_directory>/openshift** folder:

Example mce_namespace.yaml

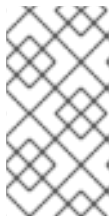
```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    openshift.io/cluster-monitoring: "true"
  name: multicluster-engine
```

Example mce_operatorgroup.yaml

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: multicluster-engine-operatorgroup
  namespace: multicluster-engine
spec:
  targetNamespaces:
    - multicluster-engine
```

Example mce_subscription.yaml

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: multicluster-engine
  namespace: multicluster-engine
spec:
  channel: "stable-2.3"
  name: multicluster-engine
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```



NOTE

You can install a distributed unit (DU) at scale with the Red Hat Advanced Cluster Management (RHACM) using the assisted installer (AI). These distributed units must be enabled in the hub cluster. The AI service requires persistent volumes (PVs), which are manually created.

- For the AI service, create the following manifests and save them in the **<assets_directory>/openshift** folder:

Example Iso_namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    openshift.io/cluster-monitoring: "true"
  name: openshift-local-storage
```

Example Iso_operatorgroup.yaml

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: local-operator-group
  namespace: openshift-local-storage
spec:
  targetNamespaces:
    - openshift-local-storage
```

Example Iso_subscription.yaml

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: local-storage-operator
  namespace: openshift-local-storage
spec:
  installPlanApproval: Automatic
  name: local-storage-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

NOTE

After creating all the manifests, your filesystem must display as follows:

Example Filesystem

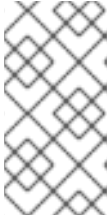
```
<assets_directory>
├── install-config.yaml
├── agent-config.yaml
├── /openshift
│   ├── mce_namespace.yaml
│   ├── mce_operatorgroup.yaml
│   ├── mce_subscription.yaml
│   ├── Iso_namespace.yaml
│   ├── Iso_operatorgroup.yaml
│   └── Iso_subscription.yaml
```

4. Create the agent ISO image by running the following command:

```
$ openshift-install agent create image --dir <assets_directory>
```

5. When the image is ready, boot the target machine and wait for the installation to complete.
6. To monitor the installation, run the following command:

```
$ openshift-install agent wait-for install-complete --dir <assets_directory>
```



NOTE

To configure a fully functional hub cluster, you must create the following manifests and manually apply them by running the command **\$ oc apply -f <manifest-name>**. The order of the manifest creation is important and where required, the waiting condition is displayed.

7. For the PVs that are required by the AI service, create the following manifests:

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: assisted-service
  namespace: openshift-local-storage
spec:
  logLevel: Normal
  managementState: Managed
  storageClassDevices:
    - devicePaths:
      - /dev/vda
      - /dev/vdb
    storageClassName: assisted-service
    volumeMode: Filesystem
```

8. Use the following command to wait for the availability of the PVs, before applying the subsequent manifests:

```
$ oc wait localvolume -n openshift-local-storage assisted-service --for condition=Available --
timeout 10m
```



NOTE

The ``devicePath`` is an example and may vary depending on the actual hardware configuration used.

9. Create a manifest for a multicluster engine instance.

Example MultiClusterEngine.yaml

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
```



```
name: multiclusterengine
spec: {}
```

10. Create a manifest to enable the AI service.

Example **agentserviceconfig.yaml**

```
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  name: agent
  namespace: assisted-installer
spec:
  databaseStorage:
    storageClassName: assisted-service
    accessModes:
      - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  filesystemStorage:
    storageClassName: assisted-service
    accessModes:
      - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

11. Create a manifest to deploy subsequently spoke clusters.

Example **clusterimageset.yaml**

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: "4.18"
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.18.0-x86_64
```

12. Create a manifest to import the agent installed cluster (that hosts the multicluster engine and the Assisted Service) as the hub cluster.

Example **autoimport.yaml**

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  labels:
    local-cluster: "true"
    cloud: auto-detect
    vendor: auto-detect
  name: local-cluster
spec:
  hubAcceptsClient: true
```

13. Wait for the managed cluster to be created.

```
$ oc wait -n multicluster-engine managedclusters local-cluster --for  
condition=ManagedClusterJoined=True --timeout 10m
```

Verification

- To confirm that the managed cluster installation is successful, run the following command:

```
$ oc get managedcluster  
NAME          HUB ACCEPTED  MANAGED CLUSTER URLS      JOINED  AVAILABLE  
AGE  
local-cluster  true          https://<your cluster url>:6443  True    True    77m
```

Additional resources

- [The Local Storage Operator](#)

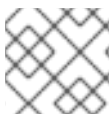
CHAPTER 8. INSTALLATION CONFIGURATION PARAMETERS FOR THE AGENT-BASED INSTALLER

Before you deploy an OpenShift Container Platform cluster using the Agent-based Installer, you provide parameters to customize your cluster and the platform that hosts it. When you create the **install-config.yaml** and **agent-config.yaml** files, you must provide values for the required parameters, and you can use the optional parameters to customize your cluster further.

8.1. AVAILABLE INSTALLATION CONFIGURATION PARAMETERS

The following tables specify the required and optional installation configuration parameters that you can set as part of the Agent-based installation process.

These values are specified in the **install-config.yaml** file.



NOTE

These settings are used for installation only, and cannot be modified after installation.

8.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

Table 8.1. 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 .
metadata:	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object

Parameter	Description	Values
metadata: name:	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}}.{{.baseDomain}} . When you do not provide metadata.name through either the install-config.yaml or agent-config.yaml files, for example when you use only ZTP manifests, the cluster name is set to agent-cluster .	String of lowercase letters, hyphens (-), and periods (.), such as dev .
platform:	The configuration for the specific platform upon which to perform the installation: baremetal , external , none , or vsphere .	Object
pullSecret:	Get a pull secret from Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.	<pre>{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }</pre>

8.1.2. Network configuration parameters

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

Consider the following information before you configure network parameters for your cluster:

- If you use the Red Hat OpenShift Networking OVN-Kubernetes network plugin, both IPv4 and IPv6 address families are supported.
- If you deployed nodes in an OpenShift Container Platform cluster with a network that supports both IPv4 and non-link-local IPv6 addresses, configure your cluster to use a dual-stack network.
 - For clusters configured for dual-stack networking, both IPv4 and IPv6 traffic must use the same network interface as the default gateway. This ensures that in a multiple network interface controller (NIC) environment, a cluster can detect what NIC to use based on the

available network interface. For more information, see "OVN-Kubernetes IPv6 and dual-stack limitations" in *About the OVN-Kubernetes network plugin*.


- To prevent network connectivity issues, do not install a single-stack IPv4 cluster on a host that supports dual-stack networking.

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


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

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

Table 8.2. Network parameters

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

Parameter	Description	Values
<code>networking: clusterNetwork:</code>	<p>The IP address blocks for pods.</p> <p>The default value is 10.128.0.0/14 with a host prefix of /23.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23 - cidr: fd01::/48 hostPrefix: 64</pre>
<code>networking: clusterNetwork: cidr:</code>	<p>Required if you use networking.clusterNetwork. An IP address block.</p> <p>If you use the OVN-Kubernetes network plugin, you can specify IPv4 and IPv6 networks.</p>	<p>An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between 0 and 32. The prefix length for an IPv6 block is between 0 and 128. For example, 10.128.0.0/14 or fd01::/48.</p>
<code>networking: clusterNetwork: hostPrefix:</code>	<p>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.</p>	<p>A subnet prefix.</p> <p>For an IPv4 network the default value is 23. For an IPv6 network the default value is 64. The default value is also the minimum value for IPv6.</p>
<code>networking: serviceNetwork:</code>	<p>The IP address block for services. The default value is 172.30.0.0/16.</p> <p>The OVN-Kubernetes network plugins supports only a single IP address block for the service network.</p> <p>If you use the OVN-Kubernetes network plugin, you can specify an IP address block for both of the IPv4 and IPv6 address families.</p>	<p>An array with an IP address block in CIDR format. For example:</p> <pre>networking: serviceNetwork: - 172.30.0.0/16 - fd02::/112</pre>
<code>networking: machineNetwork:</code>	<p>The IP address blocks for machines.</p> <p>If you specify multiple IP address blocks, the blocks must not overlap.</p>	<p>An array of objects. For example:</p> <pre>networking: machineNetwork: - cidr: 10.0.0.0/16</pre>

Parameter	Description	Values
networking: machineNetwork: cidr:	Required if you use networking.machineNetwork . An IP address block. The default value is 10.0.0.0/16 for all platforms other than libvirt and IBM Power® Virtual Server. For libvirt, the default value is 192.168.126.0/24 . For IBM Power® Virtual Server, the default value is 192.168.0.0/24 .	An IP network block in CIDR notation. For example, 10.0.0.0/16 or fd00::/48 .  <div> NOTE Set the networking.machineNetwork to match the CIDR that the preferred NIC resides in. </div>
networking: ovnKubernetesConfig: ipv4: internalJoinSubnet:	Configures the IPv4 join subnet that is used internally by ovn-kubernetes . This subnet must not overlap with any other subnet that OpenShift Container Platform is using, including the node network. The size of the subnet must be larger than the number of nodes. You cannot change the value after installation.	An IP network block in CIDR notation. The default value is 100.64.0.0/16 .


8.1.3. Optional configuration parameters


Optional installation configuration parameters are described in the following table:


Table 8.3. 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 <i>Installing</i> .	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
cpuPartitioningMode:	Enables workload partitioning, which isolates OpenShift Container Platform services, cluster management workloads, and infrastructure pods to run on a reserved set of CPUs. Workload partitioning can only be enabled during installation and cannot be disabled after installation. While this field enables workload partitioning, it does not configure workloads to use specific CPUs. For more information, see the <i>Workload partitioning</i> page in the <i>Scalability and Performance</i> section.	None or AllNodes . None is the default value.
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 , arm64 , ppc64le , and s390x .	String

Parameter	Description	Values
compute: hyperthreading:	<p>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.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
compute: name:	Required if you use compute . The name of the machine pool.	worker
compute: platform:	Required if you use compute . Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the controlPlane.platform parameter value.	baremetal , 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.

Parameter	Description	Values
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 , arm64 , ppc64le , and s390x .	String
controlPlane: hyperthreading:	<p>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.</p> <div>  <p>IMPORTANT</p> <p>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.</p> </div>	Enabled or Disabled
controlPlane: name:	Required if you use controlPlane . The name of the machine pool.	master
controlPlane: platform:	Required if you use controlPlane . Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the compute.platform parameter value.	baremetal , vsphere , or {}
controlPlane: replicas:	The number of control plane machines to provision.	Supported values are 3 , 4 , 5 , or 1 when deploying single-node OpenShift.

Parameter	Description	Values
credentialsMode:	<p>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.</p> <div>  <p>NOTE</p> <p>Not all CCO modes are supported for all cloud providers. For more information about CCO modes, see the "Managing cloud provider credentials" entry in the <i>Authentication and authorization</i> content.</p> </div>	Mint, Passthrough, Manual or an empty string ("").
fips:	<p>Enable or disable FIPS mode. The default is false (disabled). If FIPS mode is enabled, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that are provided with RHCOS instead.</p>	false or true

Parameter	Description	IMPORTANT	Values
		<p>To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode.</p> <p>When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.</p>	
imageContentSources:			Array of objects. Includes a source and, optionally, mirrors , as described in the following rows of this table.
imageContentSources: source:	Required if you use imageContentSources . Specify the repository that users refer to, for example, in image pull specifications. If you specify Azure File storage, you cannot enable FIPS mode.	NOTE	String
imageContentSources: mirrors:	Specify one or more repositories that may also contain the same images.		Array of strings

Parameter	Description	Values
 publish:	How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.	<p>Internal or External. The default value is External.</p> <p>Setting this field to Internal is not supported on non-cloud platforms.</p> <div><p>IMPORTANT</p><p>If the value of the field is set to Internal, the cluster will become non-functional. For more information, refer to BZ#1953035.</p></div>
 sshKey:	<p>The SSH key to authenticate access to your cluster machines.</p> <div><p>NOTE</p><p>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.</p></div>	For example, sshKey: ssh-ed25519 AAAA...

8.1.4. Additional bare metal configuration parameters for the Agent-based Installer

Additional bare metal installation configuration parameters for the Agent-based Installer are described in the following table:



NOTE

These fields are not used during the initial provisioning of the cluster, but they are available to use once the cluster has been installed. Configuring these fields at install time eliminates the need to set them as a Day 2 operation.

Table 8.4. Additional bare metal parameters

Parameter	Description	Values
platform: baremetal: clusterProvisioningIP:	The IP address within the cluster where the provisioning services run. Defaults to the third IP address of the provisioning subnet. For example, 172.22.0.3 or 2620:52:0:1307::3 .	IPv4 or IPv6 address.
platform: baremetal: provisioningNetwork:	<p>The provisioningNetwork configuration setting determines whether the cluster uses the provisioning network. If it does, the configuration setting also determines if the cluster manages the network.</p> <p>Managed: Default. Set this parameter to Managed to fully manage the provisioning network, including DHCP, TFTP, and so on.</p> <p>Disabled: Set this parameter to Disabled to disable the requirement for a provisioning network. When set to Disabled, you can use only virtual media based provisioning on Day 2. If Disabled and using power management, BMCs must be accessible from the bare-metal network. If Disabled, you must provide two IP addresses on the bare-metal network that are used for the provisioning services.</p>	Managed or Disabled .
platform: baremetal: provisioningMACAddress:	The MAC address within the cluster where provisioning services run.	MAC address.
platform: baremetal: provisioningNetworkCIDR:	The CIDR for the network to use for provisioning. This option is required when not using the default address range on the provisioning network.	Valid CIDR, for example 10.0.0.0/16 .

Parameter	Description	Values
platform: baremetal: provisioningNetworkInterface:	The name of the network interface on nodes connected to the provisioning network. Use the bootMACAddress configuration setting to enable IroniC to identify the IP address of the NIC instead of using the provisioningNetworkInterface configuration setting to identify the name of the NIC.	String.
platform: baremetal: provisioningDHCPRange:	Defines the IP range for nodes on the provisioning network, for example 172.22.0.10,172.22.0.254 .	IP address range.
platform: baremetal: hosts:	Configuration for bare metal hosts.	Array of host configuration objects.
platform: baremetal: hosts: name:	The name of the host.	String.
platform: baremetal: hosts: bootMACAddress:	The MAC address of the NIC used for provisioning the host.	MAC address.
platform: baremetal: hosts: bmc:	Configuration for the host to connect to the baseboard management controller (BMC).	Dictionary of BMC configuration objects.
platform: baremetal: hosts: bmc: username:	The username for the BMC.	String.

Parameter	Description	Values
<pre>platform: baremetal: hosts: bmc: password:</pre>	Password for the BMC.	String.
<pre>platform: baremetal: hosts: bmc: address:</pre>	The URL for communicating with the host's BMC controller. The address configuration setting specifies the protocol. For example, redfish+http://10.10.10.1:8000/redfish/v1/Systems/1234 enables Redfish. For more information, see "BMC addressing" in the "Deploying installer-provisioned clusters on bare metal" section.	URL.
<pre>platform: baremetal: hosts: bmc: disableCertificateVerification:</pre>	redfish and redfish-virtualmedia need this parameter to manage BMC addresses. The value should be True when using a self-signed certificate for BMC addresses.	Boolean.


8.1.5. Additional VMware vSphere configuration parameters

Additional VMware vSphere configuration parameters are described in the following table:

Table 8.5. Additional VMware vSphere cluster parameters

Parameter	Description	Values
<pre>platform: vsphere:</pre>	Describes your account on the cloud platform that hosts your cluster. You can use the parameter to customize the platform. If you provide additional configuration settings for compute and control plane machines in the machine pool, the parameter is not required.	A dictionary of vSphere configuration objects
<pre>platform: vsphere: failureDomains:</pre>	Establishes the relationships between a region and zone. You define a failure domain by using vCenter objects, such as a datastore object. A failure domain defines the vCenter location for OpenShift Container Platform cluster nodes.	An array of failure domain configuration objects.

Parameter	Description	Values
platform: vsphere: failureDomains: name:	The name of the failure domain.	String
platform: vsphere: failureDomains: region:	If you define multiple failure domains for your cluster, you must attach the tag to each vCenter data center. To define a region, use a tag from the openshift-region tag category. For a single vSphere data center environment, you do not need to attach a tag, but you must enter an alphanumeric value, such as datacenter , for the parameter.	String
platform: vsphere: failureDomains: server:	Specifies the fully-qualified hostname or IP address of the VMware vCenter server, so that a client can access failure domain resources. You must apply the server role to the vSphere vCenter server location.	String
platform: vsphere: failureDomains: zone:	If you define multiple failure domains for your cluster, you must attach a tag to each vCenter cluster. To define a zone, use a tag from the openshift-zone tag category. For a single vSphere data center environment, you do not need to attach a tag, but you must enter an alphanumeric value, such as cluster , for the parameter.	String
platform: vsphere: failureDomains: topology: computeCluster:	The path to the vSphere compute cluster.	String
platform: vsphere: failureDomains: topology: datacenter:	Lists and defines the data centers where OpenShift Container Platform virtual machines (VMs) operate. The list of data centers must match the list of data centers specified in the vcenters field.	String

Parameter	Description	Values
<pre>platform: vsphere: failureDomains: topology: datastore:</pre>	<p>The path to the vSphere datastore that holds virtual machine files, templates, and ISO images.</p>  <p>IMPORTANT</p> <p>You can specify the path of any datastore that exists in a datastore cluster. By default, Storage vMotion is automatically enabled for a datastore cluster. Red Hat does not support Storage vMotion, so you must disable Storage vMotion to avoid data loss issues for your OpenShift Container Platform cluster.</p> <p>If you must specify VMs across multiple datastores, use a datastore object to specify a failure domain in your cluster's install-config.yaml configuration file. For more information, see "VMware vSphere region and zone enablement".</p>	String
<pre>platform: vsphere: failureDomains: topology: folder:</pre>	<p>Optional: The absolute path of an existing folder where the user creates the virtual machines, for example,</p> <p>/<data_center_name>/vm/<folder_name>/<subfolder_name>.</p>	String
<pre>platform: vsphere: failureDomains: topology: networks:</pre>	<p>Lists any network in the vCenter instance that contains the virtual IP addresses and DNS records that you configured.</p>	String
<pre>platform: vsphere: failureDomains: topology: resourcePool:</pre>	<p>Optional: The absolute path of an existing resource pool where the installation program creates the virtual machines, for example,</p> <p>/<data_center_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name>.</p>	String

Parameter	Description	Values
platform: vsphere: failureDomains: topology template:	Specifies the absolute path to a pre-existing Red Hat Enterprise Linux CoreOS (RHCOS) image template or virtual machine. The installation program can use the image template or virtual machine to quickly install RHCOS on vSphere hosts. Consider using this parameter as an alternative to uploading an RHCOS image on vSphere hosts. This parameter is available for use only on installer-provisioned infrastructure.	String
platform: vsphere: vcenters:	Configures the connection details so that services can communicate with a vCenter server.	An array of vCenter configuration objects.
platform: vsphere: vcenters: datacenters:	Lists and defines the data centers where OpenShift Container Platform virtual machines (VMs) operate. The list of data centers must match the list of data centers specified in the failureDomains field.	String
platform: vsphere: vcenters: password:	The password associated with the vSphere user.	String
platform: vsphere: vcenters: port:	The port number used to communicate with the vCenter server.	Integer
platform: vsphere: vcenters: server:	The fully qualified host name (FQHN) or IP address of the vCenter server.	String
platform: vsphere: vcenters: user:	The username associated with the vSphere user.	String

8.1.6. Deprecated VMware vSphere configuration parameters

In OpenShift Container Platform 4.13, the following vSphere configuration parameters are deprecated. You can continue to use these parameters, but the installation program does not automatically specify these parameters in the **install-config.yaml** file.

The following table lists each deprecated vSphere configuration parameter:

Table 8.6. Deprecated VMware vSphere cluster parameters

Parameter	Description	Values
<code>platform: vsphere: cluster:</code>	The vCenter cluster to install the OpenShift Container Platform cluster in.	String
<code>platform: vsphere: datacenter:</code>	Defines the data center where OpenShift Container Platform virtual machines (VMs) operate.	String
<code>platform: vsphere: defaultDatastore:</code>	The name of the default datastore to use for provisioning volumes.	String
<code>platform: vsphere: folder:</code>	Optional: The absolute path of an existing folder where the installation program creates the virtual machines. If you do not provide this value, the installation program creates a folder that is named with the infrastructure ID in the data center virtual machine folder.	String, for example, <code>/<data_center_name>/vm/<folder_name>/<subfolder_name></code> .
<code>platform: vsphere: password:</code>	The password for the vCenter user name.	String
<code>platform: vsphere: resourcePool:</code>	Optional: The absolute path of an existing resource pool where the installation program creates the virtual machines. If you do not specify a value, the installation program installs the resources in the root of the cluster under <code>/<data_center_name>/host/<cluster_name>/Resources</code> .	String, for example, <code>/<data_center_name>/host/<cluster_name>/Resources/<resource_pool_name>/<optional_nested_resource_pool_name></code> .
<code>platform: vsphere: username:</code>	The user name to use to connect to the vCenter instance with. This user must have at least the roles and privileges that are required for static or dynamic persistent volume provisioning in vSphere.	String

Parameter	Description	Values
platform: vsphere: vCenter:	The fully-qualified hostname or IP address of a vCenter server.	String

Additional resources

- [BMC addressing](#)
- [Configuring regions and zones for a VMware vCenter](#)
- [Required vCenter account privileges](#)

8.2. AVAILABLE AGENT CONFIGURATION PARAMETERS

The following tables specify the required and optional Agent configuration parameters that you can set as part of the Agent-based installation process.

These values are specified in the **agent-config.yaml** file.



NOTE

These settings are used for installation only, and cannot be modified after installation.

8.2.1. Required configuration parameters

Required Agent configuration parameters are described in the following table:

Table 8.7. Required parameters

Parameter	Description	Values
apiVersion:	The API version for the agent-config.yaml content. The current version is v1beta1 . The installation program might also support older API versions.	String
metadata:	Kubernetes resource ObjectMeta , from which only the name parameter is consumed.	Object

Parameter	Description	Values
metadata: name:	The name of the cluster. DNS records for the cluster are all subdomains of {{.metadata.name}} . {{.baseDomain}} . The value entered in the agent-config.yaml file is ignored, and instead the value specified in the install-config.yaml file is used. When you do not provide metadata.name through either the install-config.yaml or agent-config.yaml files, for example when you use only ZTP manifests, the cluster name is set to agent-cluster .	String of lowercase letters and hyphens (-), such as dev .

8.2.2. Optional configuration parameters

Optional Agent configuration parameters are described in the following table:

Table 8.8. Optional parameters

Parameter	Description	Values
rendezvousIP:	The IP address of the node that performs the bootstrapping process as well as running the assisted-service component. You must provide the rendezvous IP address when you do not specify at least one host's IP address in the networkConfig parameter. If this address is not provided, one IP address is selected from the provided hosts' networkConfig .	IPv4 or IPv6 address.
bootArtifactsBaseURL:	<p>When you use the Agent-based Installer to generate a minimal ISO image, this parameter specifies a URL where the rootfs image file can be retrieved from during cluster installation. This parameter is optional for booting minimal ISO images in connected environments.</p> <p>When you use the Agent-based Installer to generate an iPXE script, this parameter specifies the URL of the server to upload Preboot Execution Environment (PXE) assets to. For more information, see "Preparing PXE assets for OpenShift Container Platform".</p>	String.

Parameter	Description	Values
additionalNTP Sources:	A list of Network Time Protocol (NTP) sources to be added to all cluster hosts, which are added to any NTP sources that are configured through other means.	List of hostnames or IP addresses.
hosts:	Host configuration. An optional list of hosts. The number of hosts defined must not exceed the total number of hosts defined in the install-config.yaml file, which is the sum of the values of the compute.replicas and controlPlane.replicas parameters.	An array of host configuration objects.
hosts: hostname:	Hostname. Overrides the hostname obtained from either the Dynamic Host Configuration Protocol (DHCP) or a reverse DNS lookup. Each host must have a unique hostname supplied by one of these methods, although configuring a hostname through this parameter is optional.	String.
hosts: interfaces:	Provides a table of the name and MAC address mappings for the interfaces on the host. If a NetworkConfig section is provided in the agent-config.yaml file, this table must be included and the values must match the mappings provided in the NetworkConfig section.	An array of host configuration objects.
hosts: interfaces: name:	The name of an interface on the host.	String.
hosts: interfaces: macAddress:	The MAC address of an interface on the host.	A MAC address such as the following example: 00-B0-D0-63-C2-26 .
hosts: role:	Defines whether the host is a master or worker node. If no role is defined in the agent-config.yaml file, roles will be assigned at random during cluster installation.	master or worker .

Parameter	Description	Values
hosts: rootDeviceHints:	Enables provisioning of the Red Hat Enterprise Linux CoreOS (RHCOS) image to a particular device. The installation program examines the devices in the order it discovers them, and compares the discovered values with the hint values. It uses the first discovered device that matches the hint value. This is the device that the operating system is written on during installation.	A dictionary of key-value pairs. For more information, see "Root device hints" in the "Setting up the environment for an OpenShift installation" page.
hosts: rootDeviceHints: deviceName:	The name of the device the RHCOS image is provisioned to.	String.
hosts: networkConfig:	The host network definition. The configuration must match the Host Network Management API defined in the nmstate documentation .	A dictionary of host network configuration objects.
minimalISO:	<p>Defines whether the Agent-based Installer generates a full ISO or a minimal ISO image. When this parameter is set to True, the Agent-based Installer generates an ISO without a rootfs image file, and instead contains details about where to pull the rootfs file from.</p> <p>When you generate a minimal ISO, if you do not specify a rootfs URL through the bootArtifactsBaseURL parameter, the Agent-based Installer embeds a default URL that is accessible in environments with an internet connection.</p> <p>The default value is False.</p>	Boolean.

Additional resources

- [Preparing PXE assets for OpenShift Container Platform](#)
- [Root device hints](#)

