Assisted Installer for OpenShift Container Platform 2022

Assisted Installer for OpenShift Container Platform

Assisted Installer User Guide

Last Updated: 2023-10-11

Assisted Installer for OpenShift Container Platform 2022 Assisted Installer for OpenShift Container Platform

Assisted Installer User Guide

Legal Notice

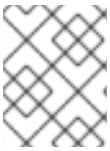
Abstract

Information about the Assisted Installer and its usage

Table of Contents

CHAPTER 1. INSTALLING AN ON-PREMISE CLUSTER USING THE ASSISTED INSTALLER

You can install OpenShift Container Platform on on-premise hardware or on-premise VMs using the Assisted Installer. Installing OpenShift Container Platform using the Assisted Installer supports **x86_64**, **ppc64le**, **s390x** and **arm64** CPU architectures.



NOTE

Currently installing OpenShift Container Platform on IBM zSystems (s390x) is only supported with RHEL KVM installations.

1.1. USING THE ASSISTED INSTALLER

The OpenShift Container Platform Assisted Installer is a user-friendly installation solution offered on the Red Hat Hybrid Cloud Console . The Assisted Installer supports the various deployment platforms with a focus on bare metal and vSphere infrastructures.

The Assisted Installer provides installation functionality as a service. This software-as-a-service (SaaS) approach has the following advantages:

- **Web user interface:** The web user interface performs cluster installation without the user having to create the installation configuration files manually.
- **No bootstrap node:** A bootstrap node is not required when installing with the Assisted Installer. The bootstrapping process executes on a node within the cluster.
- **Hosting:** The Assisted Installer hosts:
  - Ignition files
  - The installation configuration
  - A discovery ISO
  - The installer
- **Streamlined installation workflow:** Deployment does not require in-depth knowledge of OpenShift Container Platform. The Assisted Installer provides reasonable defaults and provides the installer as a service, which:
  - Eliminates the need to install and run the OpenShift Container Platform installer locally.
  - Ensures the latest version of the installer up to the latest tested z-stream releases. Older versions remain available, if needed.
  - Enables building automation by using the API without the need to run the OpenShift Container Platform installer locally.
- **Advanced networking:** The Assisted Installer supports IPv4 networking with SDN and OVN, IPv6 and dual stack networking with OVN only, NMState-based static IP addressing, and an HTTP/S proxy. OVN is the default Container Network Interface (CNI) for OpenShift Container Platform 4.12 and later releases, but you can still switch to use SDN.
- **Pre-installation validation:** The Assisted Installer validates the configuration before installation to ensure a high probability of success. Validation includes:

    - Ensuring network connectivity
    - Ensuring sufficient network bandwidth
    - Ensuring connectivity to the registry
    - Ensuring time synchronization between cluster nodes
    - Verifying that the cluster nodes meet the minimum hardware requirements
    - Validating the installation configuration parameters
- **REST API:** The Assisted Installer has a REST API, enabling automation.

The Assisted Installer supports installing OpenShift Container Platform on premises in a connected environment, including with an optional HTTP/S proxy. It can install the following:

- Highly available OpenShift Container Platform or Single Node OpenShift (SNO)
- OpenShift Container Platform on bare metal or vSphere with full platform integration, or other virtualization platforms without integration
- Optionally OpenShift Virtualization and OpenShift Data Foundation (formerly OpenShift Container Storage)

The user interface provides an intuitive interactive workflow where automation does not exist or is not required. Users may also automate installations using the REST API.

See Install OpenShift with the Assisted Installer to create an OpenShift Container Platform cluster with the Assisted Installer.

1.2. API SUPPORT FOR THE ASSISTED INSTALLER

Supported APIs for the Assisted Installer are stable for a minimum of three months from the announcement of deprecation.

CHAPTER 2. PREPARING TO INSTALL WITH THE ASSISTED INSTALLER

Before installing a cluster, you must ensure the cluster nodes and network meet the requirements.

2.1. PREREQUISITES

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- If you use a firewall, you must configure it so that Assisted Installer can access the resources it requires to function.

2.2. ASSISTED INSTALLER PREREQUISITES

The Assisted Installer validates the following prerequisites to ensure successful installation.

2.2.1. CPU Architectures

The Assisted installer is supported on the following cpu architectures:

- x86_64
- arm64
- ppc64le
- s390x

2.2.2. Hardware

The Assisted Installer requires one host with at least 8 CPU cores, 16.00 GiB RAM, and 100 GB disk size.

For the control plane, hosts must have at least the following resources:

- 4 CPU cores
- 16.00 GiB RAM
- 100 GB storage
- 10ms write speed or less for etcd **wal_fsync_duration_seconds**

For workers, each host must have at least the following resources:

- 2 CPU cores
- 8.00 GiB RAM
- 100 GB storage

2.2.3. Networking

The network must meet the following requirements:

- A DHCP server unless using static IP addressing.
- A base domain name. You must ensure that the following requirements are met:
  - There is no wildcard, such as ***.<cluster_name>.<base_domain>**, or the installation will not proceed.
  - A DNS A/AAAA record for **api.<cluster_name>.<base_domain>**.
  - A DNS A/AAAA record with a wildcard for ***.apps.<cluster_name>.<base_domain>**.
- Port **6443** is open for the API URL if you intend to allow users outside the firewall to access the cluster via the **oc** CLI tool.
- Port **443** is open for the console if you intend to allow users outside the firewall to access the console.
- A DNS A/AAAA record for each node in the cluster when using User Managed Networking, or the installation will not proceed. DNS A/AAAA records are required for each node in the cluster when using Cluster Managed Networking after installation is complete in order to connect to the cluster, but installation can proceed without the A/AAAA records when using Cluster Managed Networking.
- A DNS PTR record for each node in the cluster if you want to boot with the preset hostname when using static IP addressing. Otherwise, the Assisted Installer has an automatic node renaming feature when using static IP addressing that will rename the nodes to their network interface MAC address.

IMPORTANT

DNS A/AAAA record settings at top-level domain registrars can take significant time to update. Ensure the A/AAAA record DNS settings are working before installation to prevent installation delays.

The OpenShift Container Platform cluster's network must also meet the following requirements:

- Connectivity between all cluster nodes
- Connectivity for each node to the internet
- Access to an NTP server for time synchronization between the cluster nodes

2.2.4. Preflight validations

The Assisted Installer ensures the cluster meets the prerequisites before installation, because it eliminates complex post-installation troubleshooting, thereby saving significant amounts of time and effort. Before installing software on the nodes, the Assisted Installer conducts the following validations:

- Ensures network connectivity
- Ensures sufficient network bandwidth

- Ensures connectivity to the registry
- Ensures time synchronization between cluster nodes
- Verifies that the cluster nodes meet the minimum hardware requirements
- Validates the installation configuration parameters

If the Assisted Installer does not successfully validate the foregoing requirements, installation will not proceed.

CHAPTER 3. INSTALLING WITH THE ASSISTED INSTALLER UI

After you ensure the cluster nodes and network requirements are met, you can begin installing the cluster.

3.1. PRE-INSTALLATION CONSIDERATIONS

Before installing OpenShift Container Platform with the Assisted Installer, you must consider the following configuration choices:

- Which base domain to use
- Which OpenShift Container Platform product version to install
- Whether to install a full cluster or single-node OpenShift
- Whether to use a DHCP server or a static network configuration
- Whether to use IPv4 or dual-stack networking
- Whether to install OpenShift Virtualization
- Whether to install Red Hat OpenShift Data Foundation
- Whether to install Multicluster Engine
- Whether to integrate with the platform when installing on vSphere or Nutanix

IMPORTANT

If you intend to install any of the Operators, refer to the relevant hardware and storage requirements in Optional:Installing Operators.

3.2. SETTING THE CLUSTER DETAILS

To create a cluster with the Assisted Installer web user interface, use the following procedure.

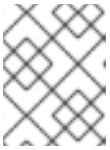
Procedure

1. Log in to the Red Hat Hybrid Cloud Console .
2. In the **Red Hat OpenShift** tile, click **Scale your applications**.
3. In the menu, click **Clusters**.
4. Click **Create cluster**.
5. Click the **Datacenter** tab.
6. Under **Assisted Installer**, click **Create cluster**.
7. Enter a name for the cluster in the **Cluster name** field.
8. Enter a base domain for the cluster in the **Base domain** field. All subdomains for the cluster will use this base domain.

NOTE

The base domain must be a valid DNS name. You must not have a wild card domain set up for the base domain.

9. Select the version of OpenShift Container Platform to install.



NOTE

For IBM Power and IBM zSystems platforms, only OpenShift Container Platform version 4.13 and later is supported.

10. Optional: Select **Install single node Openshift (SNO)** if you want to install OpenShift Container Platform on a single node.

NOTE

Currently, SNO is not supported on IBM zSystems and IBM Power platforms.

11. Optional: The Assisted Installer already has the pull secret associated to your account. If you want to use a different pull secret, select **Edit pull secret**

12. Optional: Assisted Installer defaults to using **x86_64** CPU architecture. If you are installing OpenShift Container Platform on a different architecture select the respective architecture to use. Valid values are **arm64**, **ppc64le**, and **s390x**. Keep in mind, some features are not available with **arm64**, **ppc64le**, and **s390x** CPU architectures.

13. Optional: The Assisted Installer defaults to DHCP networking. If you are using a static IP configuration, bridges or bonds for the cluster nodes instead of DHCP reservations, select **Static IP, bridges, and bonds**

14. Optional: If you want to enable encryption of the installation disks, under **Enable encryption of installation disks** you can select **Control plane node, worker** for single-node OpenShift. For multi-node clusters, you can select **Control plane nodes** to encrypt the control plane node installation disks and select **Workers** to encrypt worker node installation disks.

IMPORTANT

You cannot change the base domain, the SNO checkbox, the CPU architecture, the host's network configuration, or the disk-encryption after installation begins.

3.3. OPTIONAL: CONFIGURING STATIC NETWORKS

The Assisted Installer supports IPv4 networking with SDN and OVN, and supports IPv6 and dual stack networking with OVN only. The Assisted Installer supports configuring the network with static network interfaces with IP address/MAC address mapping. The Assisted Installer also supports configuring host network interfaces with the NMState library, a declarative network manager API for hosts. You can use NMState to deploy hosts with static IP addressing, bonds, VLANs and other advanced networking features. First, you must set network-wide configurations. Then, you must create a host-specific configuration for each host.

Procedure

1. Select the internet protocol version. Valid options are **IPv4** and **Dual stack**.
2. If the cluster hosts are on a shared VLAN, enter the VLAN ID.
3. Enter the network-wide IP addresses. If you selected **Dual stack** networking, you must enter both IPv4 and IPv6 addresses.
    a. Enter the cluster network's IP address range in CIDR notation.
    b. Enter the default gateway IP address.
    c. Enter the DNS server IP addresss.
4. Enter the host-specific configuration.
    a. If you are only setting a static IP address that uses a single network interface, use the form view to enter the IP address and the MAC address for each host.
    b. If you are using multiple interfaces, bonding, or other advanced networking features, use the YAML view and enter the desired network state for each host using NMState syntax. Then, add the MAC address and interface name for each host interface used in your network configuration.

Additional resources

- NMState version 2.1.4

3.4. CONFIGURING OPERATORS

The Assisted Installer can install with certain Operators configured. The Operators include:

- OpenShift Virtualization
- Multicluster Engine (MCE) for Kubernetes
- OpenShift Data Foundation
- Logical Volume Manager (LVM) Storage

IMPORTANT

For a detailed description of each of the Operators, together with hardware requirements, storage considerations, interdependencies, and additional installation instructions, see *Additional Resources*.

This step is optional. You can complete the installation without selecting an Operator.

Procedure

1. To install OpenShift Virtualization, select **Install OpenShift Virtualization**.
2. To install Multicluster Engine (MCE), select **Install multicluster engine**.
3. To install OpenShift Data Foundation, select **Install OpenShift Data Foundation**.
4. To install Logical Volume Manager, select **Install Logical Volume Manager**.

5. Click **Next** to proceed to the next step.

Additional resources

- Installing the OpenShift Virtualization Operator
- Installing the Multicluster Engine (MCE) Operator
- Installing the OpenShift Data Foundation Operator

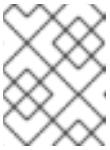
3.5. ADDING HOSTS TO THE CLUSTER

You must add one or more hosts to the cluster. Adding a host to the cluster involves generating a discovery ISO. The discovery ISO runs Red Hat Enterprise Linux CoreOS (RHCOS) in-memory with an agent.

Perform the following procedure for each host on the cluster.

Procedure

1. Click the **Add hosts** button and select the installation media.
   a. Select **Minimal image file: Provision with virtual media** to download a smaller image that will fetch the data needed to boot. The nodes must have virtual media capability. This is the recommended method for **x86_64** and **arm64** architectures.
   b. Select **Full image file: Provision with physical media** to download the larger full image. This is the recommended method for the **ppc64le** architecture.
   c. Select **iPXE: Provision from your network server** to boot the hosts using iPXE. This is the recommended method for the **s390x** architecture.



NOTE

If you install with iPXE on RHEL KVM, in some circumstances, the VMs on the KVM host are not rebooted on first boot and need to be started manually.

2. Optional: If the cluster hosts are behind a firewall that requires the use of a proxy, select **Configure cluster-wide proxy settings**. Enter the username, password, IP address and port for the HTTP and HTTPS URLs of the proxy server.
3. Add an SSH public key so that you can connect to the cluster nodes as the **core** user. Having a login to the cluster nodes can provide you with debugging information during the installation.
4. Optional: If the cluster hosts are in a network with a re-encrypting man-in-the-middle (MITM) proxy, or if the cluster needs to trust certificates for other purposes such as container image registries, select **Configure cluster-wide trusted certificates**. Add additional certificates in X.509 format.
5. Configure the discovery image if needed.
6. Optional: If you are installing on a platform and want to integrate with the platform, select **Integrate with your virtualization platform** You must boot all hosts and ensure they appear in the host inventory. All the hosts must be on the same platform.

7. Click **Generate Discovery ISO** or **Generate Script File**.
8. Download the discovery ISO or iPXE script.
9. Boot the host(s) with the discovery image or iPXE script.

Additional resources

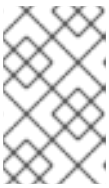
- Configuring the discovery image for additional details.
- Booting hosts with the discovery image for additional details.
- Red Hat Enterprise Linux 9 - Configuring and managing virtualization for additional details.
- How to configure a VIOS Media Repository/Virtual Media Library for additional details.
- Adding hosts on Nutanix with the UI
- Adding hosts on vSphere

3.6. CONFIGURING HOSTS

After booting the hosts with the discovery ISO, the hosts will appear in the table at the bottom of the page. You can optionally configure the hostname and role for each host. You can also delete a host if necessary.

Procedure

1. From the **Options** (⋮) menu for a host, select **Change hostname**. If necessary, enter a new name for the host and click **Change**. You must ensure that each host has a valid and unique hostname.
   Alternatively, from the **Actions** list, select **Change hostname** to rename multiple selected hosts. In the **Change Hostname** dialog, type the new name and include **{{n}}** to make each hostname unique. Then click **Change**.



NOTE

You can see the new names appearing in the **Preview** pane as you type. The name will be identical for all selected hosts, with the exception of a single-digit increment per host.

2. From the **Options** (⋮) menu, you can select **Delete host** to delete a host. Click **Delete** to confirm the deletion.
   Alternatively, from the **Actions** list, select **Delete** to delete multiple selected hosts at the same time. Then click **Delete hosts**.

NOTE

In a regular deployment, a cluster can have three or more hosts, and three of these must be control plane hosts. If you delete a host that is also a control plane, or if you are left with only two hosts, you will get a message saying that the system is not ready. To restore a host, you will need to reboot it from the discovery ISO.

3. From the **Options** (⋮) menu for the host, optionally select **View host events**. The events in the list are presented chronologically.

4. For multi-host clusters, in the **Role** column next to the host name, you can click on the menu to change the role of the host.
   If you do not select a role, the Assisted Installer will assign the role automatically. The minimum hardware requirements for control plane nodes exceed that of worker nodes. If you assign a role to a host, ensure that you assign the control plane role to hosts that meet the minimum hardware requirements.

5. Click the **Status** link to view hardware, network and operator validations for the host.

6. Click the arrow to the left of a host name to expand the host details.

Once all cluster hosts appear with a status of **Ready**, proceed to the next step.

3.7. CONFIGURING STORAGE DISKS

After discovering and configuring the cluster hosts, you can optionally configure the storage disks for each host.

Any host configurations possible here are discussed in the Configuring Hosts section. See the additional resources below for the link.

Procedure

1. To the left of the checkbox next to a host name, click to display the storage disks for that host.

2. If there are multiple storage disks for a host, you can select a different disk to act as the installation disk. Click the **Role** dropdown list for the disk, and then select **Installation disk**. The role of the previous installation disk changes to **None**.

3. All bootable disks are marked for reformatting during the installation by default, with the exception of read-only disks such as CDROMs. Deselect the **Format** checkbox to prevent a disk from being reformatted. The installation disk must be reformatted. Back up any sensitive data before proceeding.

Once all disk drives appear with a status of **Ready**, proceed to the next step.

Additional resources

- Configuring hosts

3.8. CONFIGURING NETWORKING

Before installing OpenShift Container Platform, you must configure the cluster network.

Procedure

1. In the **Networking** page, select one of the following if it is not already selected for you:

   - **Cluster-Managed Networking:** Selecting cluster-managed networking means that the Assisted Installer will configure a standard network topology, including **keepalived** and Virtual Router Redundancy Protocol (VRRP) for managing the API and Ingress VIP addresses.

NOTE

Currently, Cluster-Managed Networking is not supported on IBM zSystems and IBM Power in OpenShift Container Platform version 4.13.

- **User-Managed Networking**: Selecting user-managed networking allows you to deploy OpenShift Container Platform with a non-standard network topology. For example, if you want to deploy with an external load balancer instead of **keepalived** and VRRP, or if you intend to deploy the cluster nodes across many distinct L2 network segments.

2. For cluster-managed networking, configure the following settings:

   a. Define the **Machine network**. You can use the default network or select a subnet.

   b. Define an **API virtual IP**. An API virtual IP provides an endpoint for all users to interact with, and configure the platform.

   c. Define an **Ingress virtual IP**. An Ingress virtual IP provides an endpoint for application traffic flowing from outside the cluster.

3. For user-managed networking, configure the following settings:

   a. Select your **Networking stack type**:

      - **IPv4**: Select this type when your hosts are only using IPv4.

      - **Dual-stack**: You can select dual-stack when your hosts are using IPv4 together with IPv6.

   b. Define the **Machine network**. You can use the default network or select a subnet.

   c. Define an **API virtual IP**. An API virtual IP provides an endpoint for all users to interact with, and configure the platform.

   d. Define an **Ingress virtual IP**. An Ingress virtual IP provides an endpoint for application traffic flowing from outside the cluster.

   e. Optional: You can select **Allocate IPs via DHCP server** to automatically allocate the **API IP** and **Ingress IP** using the DHCP server.

4. Optional: Select **Use advanced networking** to configure the following advanced networking properties:

   - **Cluster network CIDR**: Define an IP address block from which Pod IP addresses are allocated.

   - **Cluster network host prefix**: Define a subnet prefix length to assign to each node.

   - **Service network CIDR**: Define an IP address to use for service IP addresses.

   - **Network type**: Select either **Software-Defined Networking (SDN)** for standard networking or **Open Virtual Networking (OVN)** for IPv6, dual-stack networking, and telco features. In OpenShift Container Platform 4.12 and later releases, OVN is the default Container Network Interface (CNI).

Additional resources

- Network configuration

3.9. PRE-INSTALLATION VALIDATION

The Assisted Installer ensures the cluster meets the prerequisites before installation, because it eliminates complex post-installation troubleshooting, thereby saving significant amounts of time and effort. Before installing the cluster, ensure the cluster and each host pass pre-installation validation.

Additional resources

- Pre-installation validation

3.10. INSTALLING THE CLUSTER

After you have completed the configuration and all the nodes are **Ready**, you can begin installation. The installation process takes a considerable amount of time, and you can monitor the installation from the Assisted Installer web console. Nodes will reboot during the installation, and they will initialize after installation.

Procedure

1. Press **Begin installation**.
2. Click on the link in the **Status** column of the **Host Inventory** list to see the installation status of a particular host.

3.11. COMPLETING THE INSTALLATION

After the cluster is installed and initialized, the Assisted Installer indicates that the installation is finished. The Assisted Installer provides the console URL, the **kubeadmin** username and password, and the **kubeconfig** file. Additionally, the Assisted Installer provides cluster details including the OpenShift Container Platform version, base domain, CPU architecture, API and Ingress IP addresses, and the cluster and service network IP addresses.

Prerequisites

- You have installed the **oc** CLI tool.

Procedure

1. Make a copy of the **kubeadmin** username and password.
2. Download the **kubeconfig** file and copy it to the **auth** directory under your working directory:

```
$ mkdir -p <working_directory>/auth
```

```
$ cp kubeadmin <working_directory>/auth
```

NOTE

The **kubeconfig** file is available for download for 24 hours after completing the installation.

3. Add the **kubeconfig** file to your environment:

```
$ export KUBECONFIG=<your working directory>/auth/kubeconfig
```

4. Login with the **oc** CLI tool:

```
$ oc login -u kubeadmin -p <password>
```

   Replace **<password>** with the password of the **kubeadmin** user.

5. Click on the web console URL or click **Launch OpenShift Console** to open the console.

6. Enter the **kubeadmin** username and password. Follow the instructions in the OpenShift Container Platform console to configure an identity provider and configure alert receivers.

7. Add a bookmark of the OpenShift Container Platform console.

8. Complete any post-installation platform integration steps.

Additional resources

- Nutanix post-installation configuration
- vSphere post-installation configuration

CHAPTER 4. INSTALLING WITH THE ASSISTED INSTALLER API

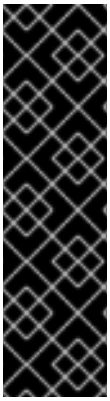
After you ensure the cluster nodes and network requirements are met, you can begin installing the cluster using the Assisted Installer API. To use the API, you must perform the following procedures:

- Set up the API authentication.
- Configure the pull secret.
- Register a new cluster definition.
- Create an infrastructure environment for the cluster.

Once you perform these steps, you can modify the cluster definition, create discovery ISOs, add hosts to the cluster, and install the cluster. This document does not cover every endpoint of the Assisted Installer API, but you can review all of the endpoints in the API viewer or the swagger.yaml file.

4.1. OPTIONAL: INSTALLING THE OPENSHIFT CLUSTER MANAGER CLI

The OpenShift Cluster Manager (ocm) CLI tool enables you to interact with the OpenShift Cluster Manager from the command line. You can execute REST GET, POST, PATCH, and DELETE operations, generate API tokens, and list clusters among other features.



IMPORTANT

OpenShift Cluster Manager CLI is a Developer Preview feature only. Developer Preview features are not supported by Red Hat in any way and are not functionally complete or production-ready. Do not use Developer Preview features for production or business-critical workloads. Developer Preview features provide early access to upcoming product features in advance of their possible inclusion in a Red Hat product offering, enabling customers to test functionality and provide feedback during the development process. These features might not have any documentation, are subject to change or removal at any time, and testing is limited. Red Hat might provide ways to submit feedback on Developer Preview features without an associated SLA.

Prerequisites

- Install **jq**.
- Log in to the OpenShift Cluster Manager as a user with cluster creation privileges.

Procedure

1. In the menu, click **OpenShift**.
2. In the submenu, click **Downloads**.
3. In the **Tokens** section under **OpenShift Cluster Manager API Token**, click **View API Token**.
4. Click **Load Token**.

IMPORTANT

Disable pop-up blockers.

5. In the **Your API token** section, copy the offline token.
6. In your terminal, set the offline token to the **OFFLINE_TOKEN** variable:

```
$ export OFFLINE_TOKEN=<copied_api_token>
```

TIP

To make the offline token permanent, add it to your profile.

7. Click **Download ocm CLI**.
8. Copy the downloaded file to your path. For example, copy the file to **/usr/bin** or **~/.local/bin** and create an **ocm** symbolic link.
9. Copy and paste the authentication command to your terminal and press **Enter** to login:

```
$ ocm login --token="${OFFLINE_TOKEN}"
```

4.2. AUTHENTICATING WITH THE REST API

API calls require authentication with the API token. Assuming you use **API_TOKEN** as a variable name, add **-H "Authorization: Bearer ${API_TOKEN}"** to API calls to authenticate with the REST API.

NOTE

The API token expires after 15 minutes.

Prerequisites

- (Optional) You have installed the OpenShift Cluster Manager (ocm) CLI tool.

Procedure

1. Set the **API_TOKEN** variable using the **OFFLINE_TOKEN** to validate the user.
   a. (Optional) On the command line terminal, execute the following command:

```
$ export API_TOKEN=$( \
  curl \
  --silent \
  --header "Accept: application/json" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "client_id=cloud-services" \
  --data-urlencode "refresh_token=${OFFLINE_TOKEN}" \
  "https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token" \
  | jq --raw-output ".access_token" \
 )
```

b. (Optional) On the command line terminal, login to the **ocm** client:

```
$ ocm login --token="${OFFLINE_TOKEN}"
```

Then, generate an API token:

```
$ export API_TOKEN=$(ocm token)
```

2. Create a script in your path for one of the token generating methods. For example:

```
$ vim ~/.local/bin/refresh-token
```

```
export API_TOKEN=$( \
  curl \
  --silent \
  --header "Accept: application/json" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "client_id=cloud-services" \
  --data-urlencode "refresh_token=${OFFLINE_TOKEN}" \
  "https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token" \
  | jq --raw-output ".access_token" \
)
```

Then, save the file.

3. Change the file mode to make it executable:

```
$ chmod +x ~/.local/bin/refresh-token
```

4. Refresh the API token:

```
$ source refresh-token
```

5. Verify that you can access the API by running the following command:

```
$ curl -s https://api.openshift.com/api/assisted-install/v2/component-versions -H
"Authorization: Bearer ${API_TOKEN}" | jq
```

Example output

```
{
  "release_tag": "v2.11.3",
  "versions": {
    "assisted-installer": "registry.redhat.io/rhai-tech-preview/assisted-installer-rhel8:v1.0.0-
211",
    "assisted-installer-controller": "registry.redhat.io/rhai-tech-preview/assisted-installer-
reporter-rhel8:v1.0.0-266",
    "assisted-installer-service": "quay.io/app-sre/assisted-service:78d113a",
    "discovery-agent": "registry.redhat.io/rhai-tech-preview/assisted-installer-agent-
```

```
rhel8:v1.0.0-195"
  }
}
```

4.3. CONFIGURING THE PULL SECRET

Many of the Assisted Installer API calls require the pull secret. Download the pull secret to a file so that you can reference it in API calls. The pull secret is a JSON object that will be included as a value within the request's JSON object. The pull secret JSON must be formatted to escape the quotes. For example:

Before

```
{"auths":{"cloud.openshift.com": ...
```

After

```
{\"auths\":{\"cloud.openshift.com\": ...
```

Procedure

1. In the menu, click **OpenShift**.
2. In the submenu, click **Downloads**.
3. In the **Tokens** section under **Pull secret**, click **Download**.
4. To use the pull secret from a shell variable, execute the following command:

```
$ export PULL_SECRET=$(cat ~/Downloads/pull-secret.txt | jq -R .)
```

5. To slurp the pull secret file using **jq**, reference it in the **pull_secret** variable, piping the value to **tojson** to ensure that it is properly formatted as escaped JSON. For example:

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
        --slurpfile pull_secret ~/Downloads/pull-secret.txt ' 1
    {
        "name": "testcluster",
        "high_availability_mode": "None",
        "openshift_version": "4.11",
        "pull_secret": $pull_secret[0] | tojson, 2
        "base_dns_domain": "example.com"
    }
')"
```

1. Slurp the pull secret file.
2. Format the pull secret to escaped JSON format.

4.4. REGISTERING A NEW CLUSTER

To register a new cluster definition with the API, use the /v2/clusters endpoint. Registering a new cluster requires the following settings:

- **name**
- **openshift-version**
- **pull_secret**
- **cpu_architecture**

See the **cluster-create-params** model in the API viewer for details on the fields you can set when registering a new cluster. When setting the **olm_operators** field, see *Additional Resources* for details on installing Operators.

After you create the cluster definition, you can modify the cluster definition and provide values for additional settings. For details on modifying a cluster, see *Additional Resources*.

Prerequisites

- You have generated a valid **API_TOKEN**. Tokens expire every 15 minutes.
- You have downloaded the pull secret.
- Optional: You have assigned the pull secret to the **$PULL_SECRET** variable.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Register a new cluster.

   a. Optional: You can register a new cluster by slurping the pull secret file in the request:

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
    --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
    "name": "testcluster",
    "openshift_version": "4.11",
    "cpu_architecture" : "<architecture_name>" (1)
    "high_availability_mode": <cluster_type>, (2)
    "base_dns_domain": "example.com",
    "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

NOTE

1 Use any of the following values: **x86_64**, **arm64**, **ppc64le**, **s390x**, **multi**.
2 Use the default value **full** to represent a multi-node OpenShift cluster, or **none** to represent a single-node OpenShift cluster.

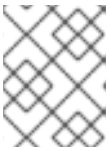
b. Optional: You can register a new cluster by writing the configuration to a JSON file and then referencing it in the request:

```
cat << EOF > cluster.json
{
  "name": "testcluster",
  "openshift_version": "4.11",
  "high_availability_mode": "<cluster_type>",
  "base_dns_domain": "example.com",
  "pull_secret": $PULL_SECRET
}
EOF
```

```
$ curl -s -X POST "https://api.openshift.com/api/assisted-install/v2/clusters" \
  -d @./cluster.json \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
  | jq '.id'
```

3. Assign the returned **cluster_id** to the **CLUSTER_ID** variable and export it:

```
$ export CLUSTER_ID=<cluster_id>
```



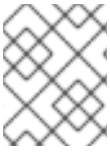
NOTE

If you close your terminal session, you need to export the **CLUSTER_ID** variable again in a new terminal session.

4. Check the status of the new cluster:

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
  | jq
```

Once you register a new cluster definition, create the infrastructure environment for the cluster.



NOTE

You cannot see the cluster configuration settings in the Assisted Installer user interface until you create the infrastructure environment.

Additional resources

- Installing Operators
- Modifying a cluster

4.5. MODIFYING A CLUSTER

To modify a cluster definition with the API, use the /v2/clusters/{cluster_id} endpoint. Modifying a cluster resource is a common operation for adding settings such as changing the network type or enabling user-managed networking. See the **v2-cluster-update-params** model in the API viewer for details on the fields you can set when modifying a cluster definition. See Installing and modifying Operators for details on defining Operators within a cluster.

Prerequisites

- You have created a new cluster resource.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Modify the cluster. For example:

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
    "ssh_public_key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDZrD4LMkAEeoU2vShhF8VM+cCZtVRgB7tqtsMx
ms2q3TOJZAgfuqReKYWm+OLOZTD+DO3Hn1pah/mU3u7uJfTUg4wEX0Le8zBu9xJVym0B
VmSFkzHfIJVTn6SfZ81NqcalisGWkpmkKXVCdnVAX6RsbHfpGKk9YPQarmRCn5KzkelJK4hrS
WpBPjdzkFXaIpf64JBZtew9XVYA3QeXkIcFuq7NBuUH9BonroPEmIXNOa41PUP1IWq3mERN
gzHZiuU8Ks/pFuU5HCMvv4qbTOIhiig7vidImHPpqYT/TCkuVi5w0ZZgkkBeLnxWxH0IdrfzgFBY
AxnpTU8Ih/4VhG538Ix1hxPaM6cXds2ic71mBbtbSrk+zjtNPaeYk1O7UpcCw4jjHspU/rVV/DY51
D5gSiiuaFPBMucnYPgUxy4FMBFfGrmGLIzTKiLzcz0DiSz1jBeTQOX++1nz+KDLBD8CPdi5k4d
q7lLkapRk85qdEvgaG5RlHMSPSS3wDrQ51fD8= user@hostname"
}
' | jq
```

4.6. REGISTERING A NEW INFRASTRUCTURE ENVIRONMENT

Once you register a new cluster definition with the Assisted Installer API, create an infrastructure environment using the v2/infra-envs endpoint. Registering a new infrastructure environment requires the following settings:

- **name**
- **pull_secret**
- **cpu_architecture**

See the **infra-env-create-params** model in the API viewer for details on the fields you can set when registering a new infrastructure environment. You can modify an infrastructure environment after you create it. As a best practice, consider including the **cluster_id** when creating a new infrastructure

environment. The **cluster_id** will associate the infrastructure environment with a cluster definition. When creating the new infrastructure environment, the Assisted Installer will also generate a discovery ISO.

Prerequisites

- You have generated a valid **API_TOKEN**. Tokens expire every 15 minutes.
- You have downloaded the pull secret.
- Optional: You have registered a new cluster definition and exported the **cluster_id**.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Register a new infrastructure environment. Provide a name, preferably something including the cluster name. This example provides the cluster ID to associate the infrastructure environment with the cluster resource. The following example specifies the **image_type**. You can specify either **full-iso** or **minimal-iso**. The default value is **minimal-iso**.

   a. Optional: You can register a new infrastructure environment by slurping the pull secret file in the request:

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt \
  --arg cluster_id ${CLUSTER_ID} '
    {
      "name": "testcluster-infra-env",
      "image_type":"full-iso",
      "cluster_id": $cluster_id,
      "cpu_architecture" : "<architecture_name>" 1
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq '.id'
```

NOTE

1 Indicates the valid values. They are: x86_64, arm64, ppc64le, s390x, multi

   b. Optional: You can register a new infrastructure environment by writing the configuration to a JSON file and then referencing it in the request:

```
$ cat << EOF > infra-envs.json
{
 "name": "testcluster-infra-env",
 "image_type": "full-iso",
 "cluster_id": "$CLUSTER_ID",
```

```
 "pull_secret": $PULL_SECRET
}
EOF
```

```
$ curl -s -X POST "https://api.openshift.com/api/assisted-install/v2/infra-envs" \
 -d @./infra-envs.json \
 -H "Content-Type: application/json" \
 -H "Authorization: Bearer $API_TOKEN" \
 | jq '.id'
```

3. Assign the returned **id** to the **INFRA_ENV_ID** variable and export it:

```
$ export INFRA_ENV_ID=<id>
```

NOTE

Once you create an infrastructure environment and associate it to a cluster definition via the **cluster_id**, you can see the cluster settings in the Assisted Installer web user interface. If you close your terminal session, you need to re-export the **id** in a new terminal session.

4.7. MODIFYING AN INFRASTRUCTURE ENVIRONMENT

You can modify an infrastructure environment using the /v2/infra-envs/{infra_env_id} endpoint. Modifying an infrastructure environment is a common operation for adding settings such as networking, SSH keys, or ignition configuration overrides.

See the **infra-env-update-params** model in the API viewer for details on the fields you can set when modifying an infrastructure environment. When modifying the new infrastructure environment, the Assisted Installer will also re-generate the discovery ISO.

Prerequisites

- You have created a new infrastructure environment.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Modify the infrastructure environment:

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
    {
      "image_type":"minimal-iso",
```

```
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq
```

4.8. ADDING HOSTS

After configuring the cluster resource and infrastructure environment, download the discovery ISO image. You can choose from two images:

- **Full ISO image:** Use the full ISO image when booting must be self-contained. The image includes everything needed to boot and start the Assisted Installer agent. The ISO image is about 1GB in size.
- **Minimal ISO image:** Use the minimal ISO image when bandwidth over the virtual media connection is limited. This is the default setting. The image includes only what is required to boot a host with networking. The majority of the content is downloaded upon boot. The ISO image is about 100MB in size.

Both images lead to the same installation procedure. To change the image type, modify the **image_type** setting in the infrastructure environment before performing this procedure.

Prerequisites

- You have created a cluster.
- You have created an infrastructure environment.
- You have completed the configuration.
- If the cluster hosts are behind a firewall that requires the use of a proxy, you have configured the username, password, IP address and port for the HTTP and HTTPS URLs of the proxy server.
- You have selected an image type or will use the default **minimal-iso**.

Procedure

1. Configure the discovery image if needed.
2. Refresh the API token:

   ```
   $ source refresh-token
   ```

3. Get the download URL:

   ```
   $ curl -H "Authorization: Bearer ${API_TOKEN}" \
   https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID}/downloads/image-url
   ```

4. Download the discovery image:

   ```
   $ wget -O discovery.iso '<url>'
   ```

   Replace **<url>** with the download URL from the previous step.

5. Boot the host(s) with the discovery image. If you are installing on a platform and want to integrate with the platform, see the additional resources below for details.

6. Assign a role to host(s).

Additional resources

- Configuring the discovery image
- Booting hosts with the discovery image
- Adding hosts on Nutanix with the API
- Adding hosts on vSphere
- Assigning roles to hosts

4.9. MODIFYING HOSTS

After adding hosts, modify the hosts as needed. The most common modifications are to the **host_name** and the **host_role** parameters.

You can modify a host using the /v2/infra-envs/{infra_env_id}/hosts/{host_id} endpoint. See the **host-update-params** model in the API viewer for details on the fields you can set when modifying a host.

A host may be one of two roles:

- **master**: A host with the **master** role will operate as a control plane host.
- **worker**: A host with the **worker** role will operate as a worker host.

By default, the Assisted Installer sets a host to **auto-assign**, which means the installer will determine whether the host is a **master** or **worker** role automatically. Use this procedure to set the host's role.

Prerequisites

- You have added hosts to the cluster.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Get the host IDs:

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

Example output

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

3. Modify the host:

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
    {
      "host_role":"worker"
      "host_name" : "worker-1"
    }
' | jq
```

1 Replace **<host_id>** with the ID of the host.

4.10. PRE-INSTALLATION VALIDATION

The Assisted Installer ensures the cluster meets the prerequisites before installation, because it eliminates complex post-installation troubleshooting, thereby saving significant amounts of time and effort. Before installing the cluster, ensure the cluster and each host pass pre-installation validation.

Additional resources

- Pre-installation validation

4.11. INSTALLING THE CLUSTER

Once the cluster hosts past validation, you can install the cluster.

Prerequisites

- You have created a cluster and infrastructure environment.
- You have added hosts to the infrastructure environment.
- The hosts have passed validation.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Install the cluster:

```
$ curl -H "Authorization: Bearer $API_TOKEN" \
-X POST \
https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/actions/install | jq
```

3. Complete any post-installation platform integration steps.

Additional resources

- Nutanix post-installation configuration
- vSphere post-installation configuration

CHAPTER 5. OPTIONAL: ENABLING DISK ENCRYPTION

You can enable encryption of installation disks using either the TPM v2 or Tang encryption modes.

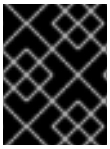
NOTE

In some situations, when you enable TPM disk encryption in the firmware for a bare-metal host and then boot it from an ISO that you generate with the Assisted Installer, the cluster deployment can get stuck. This can happen if there are left-over TPM encryption keys from a previous installation on the host. For more information, see BZ#2011634. If you experience this problem, contact Red Hat support.

5.1. ENABLING TPM V2 ENCRYPTION

Prerequisites

- Check to see if TPM v2 encryption is enabled in the BIOS on each host. Most Dell systems require this. Check the manual for your computer. The Assisted Installer will also validate that TPM is enabled in the firmware. See the **disk-encruption** model in the Assisted Installer API for additional details.



IMPORTANT

Verify that a TPM v2 encryption chip is installed on each node and enabled in the firmware.

Procedure

1. Optional: Using the UI, in the **Cluster details** step of the user interface wizard, choose to enable TPM v2 encryption on either the control plane nodes, workers, or both.

2. Optional: Using the API, follow the "Modifying hosts" procedure. Set the **disk_encryption.enable_on** setting to **all**, **masters**, or **workers**. Set the **disk_encryption.mode** setting to **tpmv2**.

   a. Refresh the API token:

   ```
   $ source refresh-token
   ```

   b. Enable TPM v2 encryption:

   ```
   $ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
   -X PATCH \
   -H "Authorization: Bearer ${API_TOKEN}" \
   -H "Content-Type: application/json" \
   -d '
   {
     "disk_encryption": {
       "enable_on": "none",
       "mode": "tpmv2"
     }
   }
   ' | jq
   ```

Valid settings for **enable_on** are **all**, **master**, **worker**, or **none**.

5.2. ENABLING TANG ENCRYPTION

Prerequisites

- You have access to a Red Hat Enterprise Linux (RHEL) 8 machine that can be used to generate a thumbprint of the Tang exchange key.

Procedure

1. Set up a Tang server or access an existing one. See Network-bound disk encryption for instructions. You can set multiple Tang servers, but the Assisted Installer must be able to connect to all of them during installation.
2. On the Tang server, retrieve the thumbprint for the Tang server using **tang-show-keys**:

   ```
   $ tang-show-keys <port>
   ```

   Optional: Replace **<port>** with the port number. The default port number is **80**.

   Example thumbprint

   ```
   1gYTN_LpU9ZMB35yn5IbADY5OQ0
   ```

3. Optional: Retrieve the thumbprint for the Tang server using **jose**.
   a. Ensure **jose** is installed on the Tang server:

      ```
      $ sudo dnf install jose
      ```

   b. On the Tang server, retrieve the thumbprint using **jose**:

      ```
      $ sudo jose jwk thp -i /var/db/tang/<public_key>.jwk
      ```

      Replace **<public_key>** with the public exchange key for the Tang server.

      Example thumbprint

      ```
      1gYTN_LpU9ZMB35yn5IbADY5OQ0
      ```

4. Optional: In the **Cluster details** step of the user interface wizard, choose to enable Tang encryption on either the control plane nodes, workers, or both. You will be required to enter URLs and thumbprints for the Tang servers.
5. Optional: Using the API, follow the "Modifying hosts" procedure.
   a. Refresh the API token:

      ```
      $ source refresh-token
      ```

b. Set the **disk_encryption.enable_on** setting to **all**, **masters**, or **workers**. Set the **disk_encryption.mode** setting to **tang**. Set **disk_encyrption.tang_servers** to provide the URL and thumbprint details about one or more Tang servers:

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "disk_encryption": {
    "enable_on": "all",
    "mode": "tang",
    "tang_servers": "
[{\"url\":\"http://tang.example.com:7500\",\"thumbprint\":\"PLjNyRdGw03zlRoGjQYMahSZGu9\"},
{\"url\":\"http://tang2.example.com:7500\",\"thumbprint\":\"XYjNyRdGw03zlRoGjQYMahSZGu3\"}]"
  }
}
' | jq
```

Valid settings for **enable_on** are **all**, **master**, **worker**, or **none**. Within the **tang_servers** value, comment out the quotes within the object(s).

5.3. ADDITIONAL RESOURCES

- Modifying hosts

CHAPTER 6. OPTIONAL: INSTALLING AND MODIFYING OPERATORS

The Assisted Installer can install select Operators for you with default configurations in either the UI or API. If you require advanced options, install the desired Operators after installing the cluster.

The Assisted Installer monitors the installation of the selected operators as part of the cluster installation and reports their status. If one or more Operators encounter errors during installation, the Assisted Installer reports that the cluster installation has completed with a warning that one or more operators failed to install.

See the sections below for the Operators you can set when installing or modifying a cluster definition using the Assisted Installer UI or API. For full instructions on installing an OpenShift Container Platform cluster, see Installing with the Assisted Installer UI or Installing with the Assisted Installer API respectively.

6.1. INSTALLING OPERATORS

When installng Operators using the Assisted Installer UI, select the Operators on the **Operators** page of the wizard. When installing Operators using the Assisted Installer API, use the POST method in the /v2/clusters endpoint.

6.1.1. Installing OpenShift Virtualization

When you configure the cluster, you can enable OpenShift Virtualization.

NOTE

Currently, OpenShift Virtualization is not supported on IBM zSystems and IBM Power.

If enabled, the Assisted Installer:

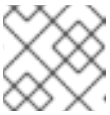
1. Validates that your environment meets the prerequisites outlined below.
2. Configures virtual machine storage as follows:
    a. For single-node OpenShift clusters version 4.10 and newer, the Assisted Installer configures the hostpath provisioner.
    b. For single-node OpenShift clusters on earlier versions, the Assisted Installer configures the Local Storage Operator.
    c. For multi-node clusters, the Assisted Installer configures OpenShift Data Foundation.

Prerequisites

- Supported by Red Hat Enterprise Linux (RHEL) 8
- Support for Intel 64 or AMD64 CPU extensions
- Intel Virtualization Technology or AMD-V hardware virtualization extensions enabled
- NX (no execute) flag enabled

Procedure

1. If you are using the Assisted Installer UI:
   - In the **Operators** step of the wizard, enable the **Install OpenShift Virtualization** checkbox.
2. If you are using the Assisted Installer API:
   - When registering a new cluster, add the **"olm_operators: [{"name": "cnv"}]"** statement.



NOTE

CNV stands for container-native virtualization.

For example:

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
  "name": "testcluster",
  "openshift_version": "4.11",
  "cpu_architecture" : "x86_64",
  "base_dns_domain": "example.com",
  "olm_operators: [{"name": "cnv"}]"
  "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

Additional resources

- For more details about preparing your cluster for OpenShift Virtualization, see the OpenShift Documentation.

6.1.2. Installing Multicluster Engine (MCE)

When you configure the cluster, you can enable the Multicluster Engine (MCE) Operator. The Multicluster Engine (MCE) Operator allows you to install additional clusters from the cluster that you are currently installing.

Prerequisites

- OpenShift version 4.10 and above
- An additional 4 CPU cores and 16GB of RAM for multi-node OpenShift clusters.
- An additional 8 CPU cores and 32GB RAM for single-node OpenShift clusters.

Storage considerations

Prior to installation, you must consider the storage required for managing the clusters to be deployed from the Multicluster Engine. You can choose one of the following scenarios for automating storage:

- Install OpenShift Data Foundation (ODF) on a multi-node cluster. ODF is the recommended storage for clusters, but requires an additional subscription. For details, see *Installing OpenShift Data Foundation* in this chapter.
- Install Logical Volume Management Storage (LVMS) on a single-node OpenShift (SNO) cluster.
- Install Multicluster Engine on a multi-node cluster without configuring storage. Then configure a storage of your choice and enable the Central Infrastructure Management (CIM) service following the installation. For details, see *Additional Resources* in this chapter.

Procedure

1. If you are using the Assisted Installer UI:
   - In the **Operators** step of the wizard, enable the **Install multicluster engine** checkbox.
2. If you are using the Assisted Installer API:
   - When registering a new cluster, use the **"olm_operators: [{"name": "mce"}]"** statement, for example:

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
    --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
    "name": "testcluster",
    "openshift_version": "4.11",
    "cpu_architecture" : "x86_64"
    "base_dns_domain": "example.com",
    "olm_operators: [{"name": "mce"}]",
    "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

Post-installation steps

- To use the Assisted Installer technology with the Multicluster Engine, enable the Central Infrastructure Management service. For details, see Enabling the Central Infrastructure Management service.
- To deploy OpenShift Container Platform clusters using hosted control planes, configure the hosted control planes. For details, see Hosted Control Planes.

Additional resources

- For Advanced Cluster Management documentation related to the Multicluster Engine (MCE) Operator, see Red Hat Advanced Cluster Mangement for Kubernetes
- For OpenShift Container Platform documentation related to the Multicluster Engine (MCE) Operator, see Multicluster Engine for Kubernetes Operator.

6.1.3. Installing OpenShift Data Foundation

When you configure the cluster, you can enable OpenShift Data Foundation. If enabled, the Assisted Installer:

1. Validates that your environment meets the prerequisites outlined below. It does not validate that the disk devices have been reformatted, which you must verify before starting.

2. Configures the storage to use all available disks.

When you enable OpenShift Data Foundation, the Assisted Installer creates a **StorageCluster** resource that specifies all available disks for use with OpenShift Data Foundation. If a different configuration is desired, modify the configuration after installing the cluster or install the Operator after the cluster is installed.

Prerequisites

- The cluster is a three-node OpenShift cluster or has at least 3 worker nodes.
- Each host has at least one non-installation disk of at least 25GB.
- The disk devices you use must be empty. There should be no Physical Volumes (PVs), Volume Groups (VGs), or Logical Volumes (LVs) remaining on the disks.
- Each host has 6 CPU cores for three-node OpenShift or 8 CPU cores for standard clusters, in addition to other CPU requirements.
- Each host has 19 GiB RAM, in addition to other RAM requirements.
- Each host has 2 CPU cores and 5GiB RAM per storage disk in addition to other CPU and RAM requirements.
- You have assigned control plane or worker roles for each host (and not auto-assign).

Procedure

1. If you are using the Assisted Installer UI:
   - In the **Operators** step of the wizard, enable the **Install OpenShift Data Foundation** checkbox.
2. If you are using the Assisted Installer API:
   - When registering a new cluster, add the **"olm_operators: [{"name": "odf"}]"** statement. For example:

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
   --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
   "name": "testcluster",
   "openshift_version": "4.11",
   "cpu_architecture" : "x86_64",
   "base_dns_domain": "example.com",
   "olm_operators: [{"name": "odf"}]",
```

```
    "pull_secret": $pull_secret[0] | tojson
  }
')" | jq '.id'
```

Additional resources

- For more details about OpenShift Data Foundation, see the OpenShift Documentation.

6.2. MODIFYING OPERATORS

In the Assisted Installer, you can add or remove Operators for a cluster resource that has already been registered as part of a previous installation step. This is only possible before you start the OpenShift Container Platform installation.

To modify the defined Operators:

- If you are using the Assisted Installer UI, navigate to the **Operators** page of the wizard and modify your selection. For details, see *Installing Operators* in this section.
- If you are using the Assisted Installer API, set the required Operator definition using the PATCH method for the /v2/clusters/{cluster_id} endpoint.

Prerequisites

- You have created a new cluster resource.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Identify the **CLUSTER_ID** variable by listing the existing clusters, as follows:

```
$ curl -s https://api.openshift.com/api/assisted-install/v2/clusters -H "Authorization: Bearer
${API_TOKEN}" | jq '[ .[] | { "name": .name, "id": .id } ]'
```

Sample output

```
[
  {
    "name": "lvmtest",
    "id": "475358f9-ed3a-442f-ab9e-48fd68bc8188" 1
  },
  {
    "name": "mcetest",
    "id": "b5259f97-be09-430e-b5eb-d78420ee509a"
  }
]
```

NOTE

1 The **id** value is the **<cluster_id>**.

3. Assign the returned **<cluster_id>** to the **CLUSTER_ID** variable and export it:

```
$ export CLUSTER_ID=<cluster_id>
```

4. Update the cluster with the new Operators:

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
    "olm_operators": [{"name": "mce"}, {"name": "cnv"}], 1
}
' | jq '.id'
```

NOTE

1 Indicates the Operators to be installed. Valid values include **mce**, **cnv**, **lvm**, and **odf**. To remove a previously installed Operator, exclude it from the list of values. To remove all previously installed Operators, type **"olm_operators": []**.

Sample output

```
{
  <various cluster properties>,
  "monitored_operators": [
    {
      "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
      "name": "console",
      "operator_type": "builtin",
      "status_updated_at": "0001-01-01T00:00:00.000Z",
      "timeout_seconds": 3600
    },
    {
      "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
      "name": "cvo",
      "operator_type": "builtin",
      "status_updated_at": "0001-01-01T00:00:00.000Z",
      "timeout_seconds": 3600
    },
    {
      "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
      "name": "mce",
      "namespace": "multicluster-engine",
      "operator_type": "olm",
      "status_updated_at": "0001-01-01T00:00:00.000Z",
      "subscription_name": "multicluster-engine",
      "timeout_seconds": 3600
    },
    {
      "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
      "name": "cnv",
```

```
    "namespace": "openshift-cnv",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "hco-operatorhub",
    "timeout_seconds": 3600
   },
   {
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "lvm",
    "namespace": "openshift-local-storage",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "local-storage-operator",
    "timeout_seconds": 4200
   }
  ],
  <more cluster properties>
```

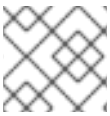
NOTE

The output is the description of the new cluster state. The **monitored_operators** property in the output contains Operators of two types:

- **"operator_type": "builtin"**: Operators of this type are an integral part of OpenShift Container Platform.
- **"operator_type": "olm"**: Operators of this type are added either manually by a user or automatically due to dependencies. In the example, the **lso** Operator was added automatically because the **cnv** Operator requires it.

CHAPTER 7. CONFIGURING THE DISCOVERY IMAGE

The Assisted Installer uses an initial image to run an agent that performs hardware and network validations before attempting to install OpenShift Container Platform. You can use Ignition to customize the discovery image.



NOTE

Modifications to the discovery image will not persist in the system.

7.1. CREATING AN IGNITION CONFIGURATION FILE

Ignition is a low-level system configuration utility, which is part of the temporary initial root filesystem, the *initramfs*. When Ignition runs on the first boot, it finds configuration data in the Ignition configuration file and applies it to the host before **switch_root** is called to pivot to the host's root filesystem.

Ignition uses a JSON configuration specification file to represent the set of changes that occur on the first boot.

IMPORTANT

Ignition versions newer than 3.2 are not supported, and will raise an error.

Procedure

1. Create an Ignition file and specify the configuration specification version:

```
$ vim ~/ignition.conf
```

```
{
  "ignition": { "version": "3.1.0" }
}
```

2. Add configuration data to the Ignition file. For example, add a password to the **core** user.

   a. Generate a password hash:

```
$ openssl passwd -6
```

   b. Add the generated password hash to the **core** user:

```
{
  "ignition": { "version": "3.1.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "passwordHash":
"$6$spam$M5LGSMGyVD.9XOboxcwrsnwNdF4irpJdAWy.1Ry55syyUiUssIzIAHaOrUHr2z
g6ruD8YNBPW9kW0H8EnKXyc1"
      }
```

```
      ]
    }
}
```

3. Save the Ignition file and export it to the **IGNITION_FILE** variable:

```
$ export IGNITION_FILE=~/ignition.conf
```

7.2. MODIFYING THE DISCOVERY IMAGE WITH IGNITION

Once you create an Ignition configuration file, you can modify the discovery image by patching the infrastructure environment using the Assisted Installer API.

Prerequisites

- If you used the UI to create the cluster, you have set up the API authentication.
- You have an infrastructure environment and you have exported the infrastructure environment **id** to the **INFRA_ENV_ID** variable.
- You have a valid Ignition file and have exported the file name as **$IGNITION_FILE**.

Procedure

1. Create an **ignition_config_override** JSON object and redirect it to a file:

```
$ jq -n \
  --arg IGNITION "$(jq -c . $IGNITION_FILE)" \
  '{ignition_config_override: $IGNITION}' \
  > discovery_ignition.json
```

2. Refresh the API token:

```
$ source refresh-token
```

3. Patch the infrastructure environment:

```
$ curl \
  --header "Authorization: Bearer $API_TOKEN" \
  --header "Content-Type: application/json" \
  -XPATCH \
  -d @discovery_ignition.json \
  https://api.openshift.com/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID | jq
```

   The **ignition_config_override** object references the Ignition file.

4. Download the updated discovery image.

CHAPTER 8. BOOTING HOSTS WITH THE DISCOVERY IMAGE

The Assisted Installer uses an initial image to run an agent that performs hardware and network validations before attempting to install OpenShift Container Platform. You can boot hosts with the discovery image using three methods:

- USB drive
- Redfish virtual media
- iPXE

8.1. CREATING AN ISO IMAGE ON A USB DRIVE

You can install the Assisted Installer agent using a USB drive that contains the discovery ISO image. Starting the host with the USB drive prepares the host for the software installation.

Procedure

1. On the administration host, insert a USB drive into a USB port.
2. Copy the ISO image to the USB drive, for example:

```
# dd if=<path_to_iso> of=<path_to_usb> status=progress
```

where:

<path_to_iso>
: is the relative path to the downloaded discovery ISO file, for example, **discovery.iso**.

<path_to_usb>
: is the location of the connected USB drive, for example, **/dev/sdb**.
After the ISO is copied to the USB drive, you can use the USB drive to install the Assisted Installer agent on the cluster host.

8.2. BOOTING WITH A USB DRIVE

To register nodes with the Assisted Installer using a bootable USB drive, use the following procedure.

Procedure

1. Insert the RHCOS discovery ISO USB drive into the target host.
2. Configure the boot drive order in the server firmware settings to boot from the attached discovery ISO, and then reboot the server.
3. Wait for the host to boot up.
   a. For UI installations, on the administration host, return to the browser. Wait for the host to appear in the list of discovered hosts.
   b. For API installations, refresh the token, check the enabled host count, and gather the host IDs:
      -

```
$ source refresh-token
```

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.enabled_host_count'
```

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

Example output

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

8.3. BOOTING FROM AN HTTP-HOSTED ISO IMAGE USING THE REDFISH API

You can provision hosts in your network using ISOs that you install using the Redfish Baseboard Management Controller (BMC) API.

Prerequisites

- Download the installation Red Hat Enterprise Linux CoreOS (RHCOS) ISO.

Procedure

1. Copy the ISO file to an HTTP server accessible in your network.
2. Boot the host from the hosted ISO file, for example:
   a. Call the redfish API to set the hosted ISO as the **VirtualMedia** boot media by running the following command:

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"Image":"<hosted_iso_file>", "Inserted": true}' \
-H "Content-Type: application/json" \
-X POST
<host_bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/Vi
rtualMedia.InsertMedia
```

Where:

<bmc_username>:<bmc_password>

Is the username and password for the target host BMC.

<hosted_iso_file>

Is the URL for the hosted installation ISO, for example: **http://webserver.example.com/rhcos-live-minimal.iso**. The ISO must be accessible from the target host machine.

<host_bmc_address>

Is the BMC IP address of the target host machine.

b. Set the host to boot from the **VirtualMedia** device by running the following command:

```
$ curl -k -u <bmc_username>:<bmc_password> \
-X PATCH -H 'Content-Type: application/json' \
-d '{"Boot": {"BootSourceOverrideTarget": "Cd", "BootSourceOverrideMode": "UEFI",
"BootSourceOverrideEnabled": "Once"}}' \
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1
```

c. Reboot the host:

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"ResetType": "ForceRestart"}' \
-H 'Content-type: application/json' \
-X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset
```

d. Optional: If the host is powered off, you can boot it using the **{"ResetType": "On"}** switch. Run the following command:

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"ResetType": "On"}' -H 'Content-type: application/json' \
-X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset
```

8.4. BOOTING HOSTS USING IPXE

The Assisted Installer provides an iPXE script including all the artifacts needed to boot the discovery image for an infrastructure environment. Due to the limitations of the current HTTPS implementation of iPXE, the recommendation is to download and expose the needed artifacts in an HTTP server. Currently, even if iPXE supports HTTPS protocol, the supported algorithms are old and not recommended.

The full list of supported ciphers is in https://ipxe.org/crypto.

Prerequisites

- You have created an infrastructure environment by using the API or you have created a cluster by using the UI.
- You have your infrastructure environment ID exported in your shell as **$INFRA_ENV_ID**.
- You have credentials to use when accessing the API and have exported a token as **$API_TOKEN** in your shell.
- You have an HTTP server to host the images.

NOTE

When configuring via the UI, the **$INFRA_ENV_ID** and **$API_TOKEN** variables are already provided.

NOTE

IBM Power only supports PXE, which also requires: **You have installed grub2 at /var/lib/tftpboot** You have installed DHCP and TFTP for PXE

Procedure

1. Download the iPXE script directly from the UI, or get the iPXE script from the Assisted Installer:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/infra-
envs/$INFRA_ENV_ID/downloads/files?file_name=ipxe-script > ipxe-script
```

Example

```
#!ipxe
initrd --name initrd http://api.openshift.com/api/assisted-images/images/<infra_env_id>/pxe-
initrd?arch=x86_64&image_token=<token_string>&version=4.10
kernel http://api.openshift.com/api/assisted-images/boot-artifacts/kernel?
arch=x86_64&version=4.10 initrd=initrd
coreos.live.rootfs_url=http://api.openshift.com/api/assisted-images/boot-artifacts/rootfs?
arch=x86_64&version=4.10 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"
boot
```

2. Download the required artifacts by extracting URLs from the **ipxe-script**.

   a. Download the initial RAM disk:

   ```
   $ awk '/^initrd /{print $NF}' ipxe-script | curl -o initrd.img
   ```

   b. Download the linux kernel:

   ```
   $ awk '/^kernel /{print $2}' ipxe-script | curl -o kernel
   ```

   c. Download the root filesystem:

   ```
   $ grep ^kernel ipxe-script | xargs -n1| grep ^coreos.live.rootfs_url | cut -d = -f 2- | curl -o
   rootfs.img
   ```

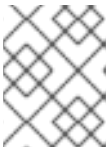
3. Change the URLs to the different artifacts in the **ipxe-script`** to match your local HTTP server. For example:

```
#!ipxe
set webserver http://192.168.0.1
```

```
initrd --name initrd $webserver/initrd.img
kernel $webserver/kernel initrd=initrd coreos.live.rootfs_url=$webserver/rootfs.img
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"
boot
```

4. Optional: When installing with RHEL KVM on IBM zSystems you must boot the host by specifying additional kernel arguments

```
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
```



NOTE

If you install with iPXE on RHEL KVM, in some circumstances, the VMs on the VM host are not rebooted on first boot and need to be started manually.

5. Optional: When installing on IBM Power you must download intramfs, kernel, and root as follows:

   a. Copy initrd.img and kernel.img to PXE directory `/var/lib/tftpboot/rhcos `

   b. Copy rootfs.img to HTTPD directory `/var/www/html/install `

   c. Add following entry to `/var/lib/tftpboot/boot/grub2/grub.cfg `:

```
if [ ${net_default_mac} == fa:1d:67:35:13:20 ]; then
default=0
fallback=1
timeout=1
menuentry "CoreOS (BIOS)" {
echo "Loading kernel"
linux "/rhcos/kernel.img" ip=dhcp rd.neednet=1 ignition.platform.id=metal ignition.firstboot
coreos.live.rootfs_url=http://9.114.98.8:8000/install/rootfs.img
echo "Loading initrd"
initrd "/rhcos/initrd.img"
}
fi
```

CHAPTER 9. ASSIGNING ROLES TO HOSTS

You can assign roles to your discovered hosts. These roles define the function of the host within the cluster. The roles can be one of the standard Kubernetes types: **control plane (master)** or **worker**.

The host must meet the minimum requirements for the role you selected. You can find the hardware requirements by referring to the Prerequisites section of this document or using the preflight requirement API.

If you do not select a role, the system selects one for you. You can change the role at any time before installation starts.

9.1. SELECT A ROLE USING THE UI

You can select a role after the host finishes its discovery.

Procedure

1. Go to the **Host Discovery** tab and scroll down to the **Host Inventory** table.
2. Select the **Auto-assign** drop-down for the required host.
    a. Select **Control plane node** to assign this host a control plane role.
    b. Select **Worker** to assign this host a worker role.
3. Check the validation status.

9.2. SELECT A ROLE USING THE API

You can select a role for the host using the /v2/infra-envs/{infra_env_id}/hosts/{host_id} endpoint. A host may be one of two roles:

- **master**: A host with the **master** role will operate as a control plane host.
- **worker**: A host with the **worker** role will operate as a worker host.

By default, the Assisted Installer sets a host to **auto-assign**, which means the installer will determine whether the host is a **master** or **worker** role automatically. Use this procedure to set the host's role.

Prerequisites

- You have added hosts to the cluster.

Procedure

1. Refresh the API token:

```
$ source refresh-token
```

2. Get the host IDs:

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
```

```
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

Example output

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

3. Modify the **host_role** setting:

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
    {
      "host_role":"worker"
    }
' | jq
```

Replace **<host_id>** with the ID of the host.

9.3. AUTO-ASSIGNING ROLES

Assisted Installer selects a role automatically for hosts if you do not assign a role yourself. The role selection mechanism factors the host's memory, CPU, and disk space. It aims to assign a control plane role to the 3 weakest hosts that meet the minimum requirements for control plane nodes. All other hosts default to worker nodes. The goal is to provide enough resources to run the control plane and reserve the more capacity-intensive hosts for running the actual workloads.

You can override the auto-assign decision at any time before installation.

The validations make sure that the auto selection is a valid one.

9.4. ADDITIONAL RESOURCES

Prerequisites

CHAPTER 10. PRE-INSTALLATION VALIDATION

10.1. DEFINITION OF PRE-INSTALLATION VALIDATIONS

The Assisted Installer aims to make cluster installation as simple, efficient, and error-free as possible. The Assisted Installer performs validation checks on the configuration and the gathered telemetry before starting an installation.

The Assisted Installer will use the information provided prior to installation, such as control plane topology, network configuration and hostnames. It will also use real time telemetry from the hosts you are attempting to install.

When a host boots the discovery ISO, an agent will start on the host. The agent will send information about the state of the host to the Assisted Installer.

The Assisted Installer uses all of this information to compute real time pre-installation validations. All validations are either blocking or non-blocking to the installation.

10.2. BLOCKING AND NON BLOCKING VALIDATIONS

A blocking validation will prevent progress of the installation, meaning that you will need to resolve the issue and pass the blocking validation before you can proceed.

A non blocking validation is a warning and will tell you of things that might cause you a problem.

10.3. VALIDATION TYPES

The Assisted Installer performs two types of validation:

Host

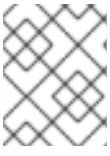
Host validations ensure that the configuration of a given host is valid for installation.

Cluster

Cluster validations ensure that the configuration of the whole cluster is valid for installation.

10.4. HOST VALIDATIONS

10.4.1. Getting host validations by using the REST API



NOTE

If you use the web based UI, many of these validations will not show up by name. To get a list of validations consistent with the labels, use the following procedure.

Prerequisites

- You have installed the **jq** utility.
- You have created an Infrastructure Environment by using the API or have created a cluster by using the UI.

- You have hosts booted with the discovery ISO
- You have your Cluster ID exported in your shell as **CLUSTER_ID**.
- You have credentials to use when accessing the API and have exported a token as **API_TOKEN** in your shell.

Procedures

1. Refresh the API token:

```
$ source refresh-token
```

2. Get all validations for all hosts:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/hosts \
  | jq -r .[].validations_info \
  | jq 'map(.[])'
```

3. Get non-passing validations for all hosts:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/hosts \
  | jq -r .[].validations_info \
  | jq 'map(.[]) | map(select(.status=="failure" or .status=="pending")) | select(length>0)'
```

10.4.2. Host validations in detail

Parameter	Validation type	Description
connected	non-blocking	Checks that the host has recently communicated with the Assisted Installer.
has-inventory	non-blocking	Checks that the Assisted Installer received the inventory from the host.
has-min-cpu-cores	non-blocking	Checks that the number of CPU cores meets the minimum requirements.
has-min-memory	non-blocking	Checks that the amount of memory meets the minimum requirements.
has-min-valid-disks	non-blocking	Checks that at least one available disk meets the eligibility criteria.

Parameter	Validation type	Description
has-cpu-cores-for-role	blocking	Checks that the number of cores meets the minimum requirements for the host role.
has-memory-for-role	blocking	Checks that the amount of memory meets the minimum requirements for the host role.
ignition-downloadable	blocking	For day 2 hosts, checks that the host can download ignition configuration from the day 1 cluster.
belongs-to-majority-group	blocking	The majority group is the largest full-mesh connectivity group on the cluster, where all members can communicate with all other members. This validation checks that hosts in a multi-node, day 1 cluster are in the majority group.
valid-platform-network-settings	blocking	Checks that the platform is valid for the network settings.
ntp-synced	non-blocking	Checks if an NTP server has been successfully used to synchronize time on the host.
container-images-available	non-blocking	Checks if container images have been successfully pulled from the image registry.
sufficient-installation-disk-speed	blocking	Checks that disk speed metrics from an earlier installation meet requirements, if they exist.
sufficient-network-latency-requirement-for-role	blocking	Checks that the average network latency between hosts in the cluster meets the requirements.
sufficient-packet-loss-requirement-for-role	blocking	Checks that the network packet loss between hosts in the cluster meets the requirements.
has-default-route	blocking	Checks that the host has a default route configured.
api-domain-name-resolved-correctly	blocking	For a multi node cluster with user managed networking. Checks that the host is able to resolve the API domain name for the cluster.
api-int-domain-name-resolved-correctly	blocking	For a multi node cluster with user managed networking. Checks that the host is able to resolve the internal API domain name for the cluster.
apps-domain-name-resolved-correctly	blocking	For a multi node cluster with user managed networking. Checks that the host is able to resolve the internal apps domain name for the cluster.

Parameter	Validation type	Description
compatible-with-cluster-platform	non-blocking	Checks that the host is compatible with the cluster platform
dns-wildcard-not-configured	blocking	Checks that the wildcard DNS *.<cluster_name>.<base_domain> is not configured, because this causes known problems for OpenShift
disk-encryption-requirements-satisfied	non-blocking	Checks that the type of host and disk encryption configured meet the requirements.
non-overlapping-subnets	blocking	Checks that this host does not have any overlapping subnets.
hostname-unique	blocking	Checks that the hostname is unique in the cluster.
hostname-valid	blocking	Checks the validity of the hostname, meaning that it matches the general form of hostnames and is not forbidden.
belongs-to-machine-cidr	blocking	Checks that the host IP is in the address range of the machine CIDR.
lso-requirements-satisfied	blocking	Validates that the cluster meets the requirements of the Local Storage Operator.
odf-requirements-satisfied	blocking	Validates that the cluster meets the requirements of the Openshift Data Foundation Operator. • The cluster has a minimum of 3 hosts. • The cluster has only 3 masters or a minimum of 3 workers. • The cluster has 3 eligible disks and each host must have an eligible disk. • The host role must not be "Auto Assign" for clusters with more than three hosts.

Parameter	Validation type	Description
cnv-requirements-satisfied	blocking	Validates that the cluster meets the requirements of Container Native Virtualization. • The BIOS of the host must have CPU virtualization enabled. • Host must have enough CPU cores and RAM available for Container Native Virtualization. • Will validate the Host Path Provisioner if necessary.
lvm-requirements-satisfied	blocking	Validates that the cluster meets the requirements of the Logical Volume Manager Operator. • Host has at least one additional empty disk, not partitioned and not formatted.
vsphere-disk-uuid-enabled	non-blocking	Verifies that each valid disk sets *disk.EnableUUID* to *true*. In VSphere this will result in each disk having a UUID.
compatible-agent	blocking	Checks that the discovery agent version is compatible with the agent docker image version.
no-skip-installation-disk	blocking	Checks that installation disk is not skipping disk formatting.
no-skip-missing-disk	blocking	Checks that all disks marked to skip formatting are in the inventory. A disk ID can change on reboot, and this validation prevents issues caused by that.
media-connected	blocking	Checks the connection of the installation media to the host.
machine-cidr-defined	non-blocking	Checks that the machine network definition exists for the cluster.
id-platform-network-settings	blocking	Checks that the platform is compatible with the network settings. Some platforms are only permitted when installing Single Node Openshift or when using User Managed Networking.

10.5. CLUSTER VALIDATIONS

10.5.1. Getting cluster validations by using the REST API

Note: If you use the web based UI, many of these validations will not show up by name. To get a list of validations consistent with the labels, use the following procedure.

Prerequisites

- You have installed the **jq** utility.
- You have created an Infrastructure Environment by using the API or have created a cluster by using the UI.
- You have your Cluster ID exported in your shell as **CLUSTER_ID**.
- You have credentials to use when accessing the API and have exported a token as **API_TOKEN** in your shell.

Procedures

1. Refresh the API token:

```
$ source refresh-token
```

2. Get all cluster validations:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID \
  | jq -r .validations_info \
  | jq 'map(.[])'
```

3. Get non-passing cluster validations:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID \
  | jq -r .validations_info \
  | jq '. | map(.[] | select(.status=="failure" or .status=="pending")) | select(length>0)'
```

10.5.2. Cluster validations in detail

Parameter	Validation type	Description
machine-cidr-defined	non-blocking	Checks that the machine network definition exists for the cluster.
cluster-cidr-defined	non-blocking	Checks that the cluster network definition exists for the cluster.
service-cidr-defined	non-blocking	Checks that the service network definition exists for the cluster.

Parameter	Validation type	Description
no-cidrs-overlapping	blocking	Checks that the defined networks do not overlap.
networks-same-address-families	blocking	Checks that the defined networks share the same address families (valid address families are IPv4, IPv6)
network-prefix-valid	blocking	Checks the cluster network prefix to ensure that it is valid and allows enough address space for all hosts.
machine-cidr-equals-to-calculated-cidr	blocking	For a non user managed networking cluster. Checks that **apiVIPs** or **ingressVIPs** are members of the machine CIDR if they exist.
api-vips-defined	non-blocking	For a non user managed networking cluster. Checks that **apiVIPs** exist.
api-vips-valid	blocking	For a non user managed networking cluster. Checks if the **apiVIPs** belong to the machine CIDR and are not in use.
ingress-vips-defined	blocking	For a non user managed networking cluster. Checks that **ingressVIPs** exist.
ingress-vips-valid	non-blocking	For a non user managed networking cluster. Checks if the **ingressVIPs** belong to the machine CIDR and are not in use.
all-hosts-are-ready-to-install	blocking	Checks that all hosts in the cluster are in the "ready to install" status.
sufficient-masters-count	blocking	This validation only applies to multi-node clusters. • The cluster must have exactly three masters. • If the cluster has worker nodes, a minimum of 2 worker nodes must exist.
dns-domain-defined	non-blocking	Checks that the base DNS domain exists for the cluster.
pull-secret-set	non-blocking	Checks that the pull secret exists. Does not check that the pull secret is valid or authorized.
ntp-server-configured	blocking	Checks that each of the host clocks are no more than 4 minutes out of sync with each other.

Parameter	Validation type	Description
lso-requirements-satisfied	blocking	Validates that the cluster meets the requirements of the Local Storage Operator.
odf-requirements-satisfied	blocking	Validates that the cluster meets the requirements of the Openshift Data Foundation Operator. ● The cluster has a minimum of 3 hosts. ● The cluster has only 3 masters or a minimum of 3 workers. ● The cluster has 3 eligible disks and each host must have an eligible disk.
cnv-requirements-satisfied	blocking	Validates that the cluster meets the requirements of Container Native Virtualization. ● The CPU architecture for the cluster is x86
lvm-requirements-satisfied	blocking	Validates that the cluster meets the requirements of the Logical Volume Manager Operator. ● The cluster must be single node. ● The cluster must be running Openshift >= 4.11.0.
network-type-valid	blocking	Checks the validity of the network type if it exists. ● The network type must be OpenshiftSDN or OVNKubernetes. ● OpenshiftSDN does not support IPv6 or Single Node Openshift. ● OVNKubernetes does not support VIP DHCP allocation.

CHAPTER 11. NETWORK CONFIGURATION

This section describes the basics of network configuration using the Assisted Installer.

11.1. CLUSTER NETWORKING

There are various network types and addresses used by OpenShift and listed in the table below.

Type	DNS	Description
clusterNetwork		The IP address pools from which Pod IP addresses are allocated.
serviceNetwork		The IP address pool for services.
machineNetwork		The IP address blocks for machines forming the cluster.
apiVIP	**api.<clustername.clusterdomain>**	The VIP to use for API communication. This setting must either be provided or pre-configured in the DNS so that the default name resolves correctly. If you are deploying with dual-stack networking, this must be the IPv4 address.
apiVIPs	**api.<clustername.clusterdomain>**	The VIPs to use for API communication. This setting must either be provided or pre-configured in the DNS so that the default name resolves correctly. If using dual stack networking, the first address must be the IPv4 address and the second address must be the IPv6 address. You must also set the **apiVIP** setting.
ingressVIP	***.apps.<clustername.clusterdomain>**	The VIP to use for ingress traffic. If you are deploying with dual-stack networking, this must be the IPv4 address.
ingressVIPs	***.apps.<clustername.clusterdomain>**	The VIPs to use for ingress traffic. If you are deploying with dual-stack networking, the first address must be the IPv4 address and the second address must be the IPv6 address. You must also set the **ingressVIP** setting.

NOTE

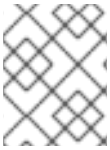
OpenShift Container Platform 4.12 introduces the new **apiVIPs** and **ingressVIPs** settings to accept multiple IP addresses for dual-stack networking. When using dual-stack networking, the first IP address must be the IPv4 address and the second IP address must be the IPv6 address. The new settings will replace **apiVIP** and **IngressVIP**, but you must set both the new and old settings when modifying the configuration using the API.

Depending on the desired network stack, you can choose different network controllers. Currently, the Assisted Service can deploy OpenShift Container Platform clusters using one of the following configurations:

- IPv4
- IPv6
- Dual-stack (IPv4 + IPv6)

Supported network controllers depend on the selected stack and are summarized in the table below. For a detailed Container Network Interface (CNI) network provider feature comparison, refer to the OCP Networking documentation.

Stack	SDN	OVN
IPv4	Yes	Yes
IPv6	No	Yes
Dual-stack	No	Yes



NOTE

OVN is the default Container Network Interface (CNI) in OpenShift Container Platform 4.12 and later releases.

11.1.1. Limitations

11.1.1.1. SDN

- With Single Node OpenShift (SNO), the SDN controller is not supported.
- The SDN controller does not support IPv6.

11.1.1.2. OVN-Kubernetes

Please see the OVN-Kubernetes limitations section in the OCP documentation .

11.1.2. Cluster network

The cluster network is a network from which every Pod deployed in the cluster gets its IP address. Given that the workload may live across many nodes forming the cluster, it's important for the network provider to be able to easily find an individual node based on the Pod's IP address. To do this,

clusterNetwork.cidr is further split into subnets of the size defined in **clusterNetwork.hostPrefix**.

The host prefix specifies a length of the subnet assigned to each individual node in the cluster. An example of how a cluster may assign addresses for the multi-node cluster:

```
---
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
---
```

Creating a 3-node cluster using the snippet above may create the following network topology:

- Pods scheduled in node #1 get IPs from **10.128.0.0/23**
- Pods scheduled in node #2 get IPs from **10.128.2.0/23**
- Pods scheduled in node #3 get IPs from **10.128.4.0/23**

Explaining OVN-K8s internals is out of scope for this document, but the pattern described above provides a way to route Pod-to-Pod traffic between different nodes without keeping a big list of mapping between Pods and their corresponding nodes.

11.1.3. Machine network

The machine network is a network used by all the hosts forming the cluster to communicate with each other. This is also the subnet that must include the API and Ingress VIPs.

11.1.4. SNO compared to multi-node cluster

Depending on whether you are deploying a Single Node OpenShift or a multi-node cluster, different values are mandatory. The table below explains this in more detail.

Parameter	SNO	Multi-Node Cluster with DHCP mode	Multi-Node Cluster without DHCP mode
clusterNetwork	Required	Required	Required
serviceNetwork	Required	Required	Required
machineNetwork	Auto-assign possible (*)	Auto-assign possible (*)	Auto-assign possible (*)
apiVIP	Forbidden	Forbidden	Required
apiVIPs	Forbidden	Forbidden	Required in 4.12 and later releases
ingressVIP	Forbidden	Forbidden	Required
ingressVIPs	Forbidden	Forbidden	Required in 4.12 and later releases

(*) Auto assignment of the machine network CIDR happens if there is only a single host network. Otherwise you need to specify it explicitly.

11.1.5. Air-gapped environments

The workflow for deploying a cluster without Internet access has some prerequisites which are out of scope of this document. You may consult the Zero Touch Provisioning the hard way Git repository for some insights.

11.2. DHCP VIP ALLOCATION

The VIP DHCP allocation is a feature allowing users to skip the requirement of manually providing virtual IPs for API and Ingress by leveraging the ability of a service to automatically assign those IP addresses from the DHCP server.

If you enable the feature, instead of using **api_vips** and **ingress_vips** from the cluster configuration, the service will send a lease allocation request and based on the reply it will use VIPs accordingly. The service will allocate the IP addresses from the Machine Network.

Please note this is not an OpenShift Container Platform feature and it has been implemented in the Assisted Service to make the configuration easier.

11.2.1. Example payload to enable autoallocation

```
---
{
  "vip_dhcp_allocation": true,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    }
  ],
  "service_networks": [
    {
      "cidr": "172.30.0.0/16"
    }
  ],
  "machine_networks": [
    {
      "cidr": "192.168.127.0/24"
    }
  ]
}
---
```

11.2.2. Example payload to disable autoallocation

```
---
{
  "api_vip": "192.168.127.100",
  "api_vips": [
```

```
    {
      "ip": "192.168.127.100"
    }
  ],
  "ingress_vip": "192.168.127.101",
  "ingress_vips": [
    {
      "ip": "192.168.127.101"
    }
  ],
  "vip_dhcp_allocation": false,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    }
  ],
  "service_networks": [
    {
      "cidr": "172.30.0.0/16"
    }
  ]
}
---
```

NOTE

In OpenShift Container Platform 4.12, you must set both the legacy **api_vip** and **ingress_vip** settings, and the new **api_vips** and **ingress_vips** settings.

11.3. ADDITIONAL RESOURCES

- Bare metal IPI documentation provides additional explanation of the syntax for the VIP addresses.

11.4. UNDERSTANDING DIFFERENCES BETWEEN USER MANAGED NETWORKING AND CLUSTER MANAGED NETWORKING

User managed networking is a feature in the Assisted Installer that allows customers with non-standard network topologies to deploy OpenShift Container Platform clusters. Examples include:

- Customers with an external load balancer who do not want to use **keepalived** and VRRP for handling VIP addressses.
- Deployments with cluster nodes distributed across many distinct L2 network segments.

11.4.1. Validations

There are various network validations happening in the Assisted Installer before it allows the installation to start. When you enable User Managed Networking, the following validations change:

- L3 connectivity check (ICMP) is performed instead of L2 check (ARP)

11.5. STATIC NETWORK CONFIGURATION

You may use static network configurations when generating or updating the discovery ISO.

11.5.1. Prerequisites

- You are familiar with NMState.

11.5.2. NMState configuration

The NMState file in YAML format specifies the desired network configuration for the host. It has the logical names of the interfaces that will be replaced with the actual name of the interface at discovery time.

11.5.2.1. Example of NMState configuration

```
---
dns-resolver:
  config:
    server:
    - 192.168.126.1
interfaces:
- ipv4:
    address:
    - ip: 192.168.126.30
      prefix-length: 24
    dhcp: false
    enabled: true
  name: eth0
  state: up
  type: ethernet
- ipv4:
    address:
    - ip: 192.168.141.30
      prefix-length: 24
    dhcp: false
    enabled: true
  name: eth1
  state: up
  type: ethernet
routes:
  config:
  - destination: 0.0.0.0/0
    next-hop-address: 192.168.126.1
    next-hop-interface: eth0
    table-id: 254
---
```

11.5.3. MAC interface mapping

MAC interface map is an attribute that maps logical interfaces defined in the NMState configuration with the actual interfaces present on the host.

The mapping should always use physical interfaces present on the host. For example, when the NMState configuration defines a bond or VLAN, the mapping should only contain an entry for parent interfaces.

11.5.3.1. Example of MAC interface mapping

```
---
mac_interface_map: [
    {
      mac_address: 02:00:00:2c:23:a5,
      logical_nic_name: eth0
    },
    {
      mac_address: 02:00:00:68:73:dc,
      logical_nic_name: eth1
    }
]
---
```

11.5.4. Additional NMState configuration examples

The examples below are only meant to show a partial configuration. They are not meant to be used as-is, and you should always adjust to the environment where they will be used. If used incorrectly, they may leave your machines with no network connectivity.

11.5.4.1. Tagged VLAN

```
---
    interfaces:
    - ipv4:
        address:
        - ip: 192.168.143.15
          prefix-length: 24
        dhcp: false
        enabled: true
      ipv6:
        enabled: false
      name: eth0.404
      state: up
      type: vlan
      vlan:
        base-iface: eth0
        id: 404
---
```

11.5.4.2. Network bond

```
---
    interfaces:
    - ipv4:
        address:
        - ip: 192.168.138.15
          prefix-length: 24
        dhcp: false
```

```
      enabled: true
    ipv6:
      enabled: false
    link-aggregation:
      mode: active-backup
      options:
        all_slaves_active: delivered
        miimon: "140"
      slaves:
      - eth0
      - eth1
    name: bond0
    state: up
    type: bond
---
```

11.6. APPLYING A STATIC NETWORK CONFIGURATION WITH THE API

You can apply a static network configuration using the Assisted Installer API.

Prerequisites

1. You have created an infrastructure environment using the API or have created a cluster using the UI.
2. You have your infrastructure environment ID exported in your shell as **$INFRA_ENV_ID**.
3. You have credentials to use when accessing the API and have exported a token as **$API_TOKEN** in your shell.
4. You have YAML files with a static network configuration available as **server-a.yaml** and **server-b.yaml**.

Procedure

1. Create a temporary file **/tmp/request-body.txt** with the API request:

```
---
jq -n --arg NMSTATE_YAML1 "$(cat server-a.yaml)" --arg NMSTATE_YAML2 "$(cat server-b.yaml)" \
'{
  "static_network_config": [
    {
      "network_yaml": $NMSTATE_YAML1,
      "mac_interface_map": [{"mac_address": "02:00:00:2c:23:a5", "logical_nic_name": "eth0"},
{"mac_address": "02:00:00:68:73:dc", "logical_nic_name": "eth1"}]
    },
    {
      "network_yaml": $NMSTATE_YAML2,
      "mac_interface_map": [{"mac_address": "02:00:00:9f:85:eb", "logical_nic_name": "eth1"},
{"mac_address": "02:00:00:c8:be:9b", "logical_nic_name": "eth0"}]
    }
  ]
}' >> /tmp/request-body.txt
---
```

2. Refresh the API token:

```
$ source refresh-token
```

3. Send the request to the Assisted Service API endpoint:

```
---
curl -H "Content-Type: application/json" \
-X PATCH -d @/tmp/request-body.txt \
-H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID
---
```

11.7. ADDITIONAL RESOURCES

- Applying a static network configuration with the UI

11.8. CONVERTING TO DUAL-STACK NETWORKING

Dual-stack IPv4/IPv6 configuration allows deployment of a cluster with pods residing in both IPv4 and IPv6 subnets.

11.8.1. Prerequisites

- You are familiar with OVN-K8s documentation

11.8.2. Example payload for Single Node OpenShift

```
---
{
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    },
    {
      "cidr": "fd01::/48",
      "host_prefix": 64
    }
  ],
  "service_networks": [
    {"cidr": "172.30.0.0/16"}, {"cidr": "fd02::/112"}
  ],
  "machine_networks": [
    {"cidr": "192.168.127.0/24"},{"cidr": "1001:db8::/120"}
  ]
}
---
```

11.8.3. Example payload for an OpenShift Container Platform cluster consisting of many nodes

```
---
{
  "vip_dhcp_allocation": false,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "api_vip": "192.168.127.100",
  "api_vips": [
    {
      "ip": "192.168.127.100"
    },
    {
      "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
    }
  ],
  "ingress_vip": "192.168.127.101",
  "ingress_vips": [
    {
      "ip": "192.168.127.101"
    },
    {
      "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7335"
    }
  ],
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    },
    {
      "cidr": "fd01::/48",
      "host_prefix": 64
    }
  ],
  "service_networks": [
    {"cidr": "172.30.0.0/16"}, {"cidr": "fd02::/112"}
  ],
  "machine_networks": [
    {"cidr": "192.168.127.0/24"},{"cidr": "1001:db8::/120"}
  ]
}
---
```

11.8.4. Limitations

The **api_vip** IP address and **ingress_vip** IP address settings must be of the primary IP address family when using dual-stack networking, which must be IPv4 addresses. Currently, Red Hat does not support dual-stack VIPs or dual-stack networking with IPv6 as the primary IP address family. Red Hat supports dual-stack networking with IPv4 as the primary IP address family and IPv6 as the secondary IP address family. Therefore, you must place the IPv4 entries before the IPv6 entries when entering the IP address values.

In OpenShift Container Platform 4.12, you must set both the legacy **api_vip** and **ingress_vip** settings,

and the new **api_vips** and **ingress_vips** settings if you want to use dual stack networking on the VIP addresses. Since the legacy **api_vip** and **ingress_vip** settings only take one value each, they must be IPv4 addresses.

11.9. ADDITIONAL RESOURCES

- Understanding OpenShift networking
- OpenShift SDN - CNI network provider
- OVN-Kubernetes - CNI network provider
- Dual-stack Service configuration scenarios
- Installing on bare metal OCP .
- Cluster Network Operator configuration .

CHAPTER 12. EXPANDING THE CLUSTER

You can expand a cluster installed with the Assisted Installer by adding hosts using the user interface or the API.

12.1. PREREQUISITES

- You must have access to an Assisted Installer cluster.
- You must install the OpenShift CLI (**oc**).
- Ensure that all the required DNS records exist for the cluster that you are adding the worker node to.
- If you are adding a worker node to a cluster with multiple CPU architectures, you must ensure that the architecture is set to **multi**.

12.2. CHECKING FOR MULTIPLE ARCHITECTURES

When adding a node to a cluster with multiple architectures, ensure that the **architecture** setting is set to **multi**.

Procedure

1. Log in to the cluster using the CLI.
2. Check the **architecture** setting:

```
$ oc adm release info -o json | jq .metadata.metadata
```

Ensure that the **architecture** setting is set to 'multi'.

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

12.3. ADDING HOSTS WITH THE UI

You can add hosts to clusters that were created using the Assisted Installer.

IMPORTANT

Adding hosts to Assisted Installer clusters is only supported for clusters running OpenShift Container Platform version 4.11 and up.

Procedure

1. Log in to OpenShift Cluster Manager and click the cluster that you want to expand.
2. Click **Add hosts** and download the discovery ISO for the new host, adding an SSH public key and configuring cluster-wide proxy settings as needed.
3. Optional: Modify ignition files as needed.

4. Boot the target host using the discovery ISO, and wait for the host to be discovered in the console.
5. Select the host role. It can be either a **worker** or a **control plane** host.
6. Start the installation.
7. As the installation proceeds, the installation generates pending certificate signing requests (CSRs) for the host. When prompted, approve the pending CSRs to complete the installation. When the host is sucessfully installed, it is listed as a host in the cluster web console.

IMPORTANT

New hosts will be encrypted using the same method as the original cluster.

12.4. ADDING HOSTS WITH THE API

You can add hosts to clusters using the Assisted Installer REST API.

Prerequisites

- Install the OpenShift Cluster Manager CLI (**ocm**).
- Log in to OpenShift Cluster Manager as a user with cluster creation privileges.
- Install **jq**.
- Ensure that all the required DNS records exist for the cluster that you want to expand.

Procedure

1. Authenticate against the Assisted Installer REST API and generate an API token for your session. The generated token is valid for 15 minutes only.
2. Set the **$API_URL** variable by running the following command:

```
$ export API_URL=<api_url> 1
```

1 Replace **<api_url>** with the Assisted Installer API URL, for example, **https://api.openshift.com**

3. Import the cluster by running the following commands:
   a. Set the **$CLUSTER_ID** variable. Log in to the cluster and run the following command:

```
$ export CLUSTER_ID=$(oc get clusterversion -o jsonpath='{.items[].spec.clusterID}')
```

   b. Set the **$CLUSTER_REQUEST** variable that is used to import the cluster:

```
$ export CLUSTER_REQUEST=$(jq --null-input --arg openshift_cluster_id
"$CLUSTER_ID" '{
  "api_vip_dnsname": "<api_vip>", 1
```

```
  "openshift_cluster_id": $CLUSTER_ID,
  "name": "<openshift_cluster_name>" 2
}')
```

1. Replace **<api_vip>** with the hostname for the cluster's API server. This can be the DNS domain for the API server or the IP address of the single node which the host can reach. For example, **api.compute-1.example.com**.

2. Replace **<openshift_cluster_name>** with the plain text name for the cluster. The cluster name should match the cluster name that was set during the Day 1 cluster installation.

c. Import the cluster and set the **$CLUSTER_ID** variable. Run the following command:

```
$ CLUSTER_ID=$(curl "$API_URL/api/assisted-install/v2/clusters/import" -H
"Authorization: Bearer ${API_TOKEN}" -H 'accept: application/json' -H 'Content-Type:
application/json' \
  -d "$CLUSTER_REQUEST" | tee /dev/stderr | jq -r '.id')
```

4. Generate the **InfraEnv** resource for the cluster and set the **$INFRA_ENV_ID** variable by running the following commands:

a. Download the pull secret file from Red Hat OpenShift Cluster Manager at console.redhat.com.

b. Set the **$INFRA_ENV_REQUEST** variable:

```
export INFRA_ENV_REQUEST=$(jq --null-input \
    --slurpfile pull_secret <path_to_pull_secret_file> \ 1
    --arg ssh_pub_key "$(cat <path_to_ssh_pub_key>)" \ 2
    --arg cluster_id "$CLUSTER_ID" '{
  "name": "<infraenv_name>", 3
  "pull_secret": $pull_secret[0] | tojson,
  "cluster_id": $cluster_id,
  "ssh_authorized_key": $ssh_pub_key,
  "image_type": "<iso_image_type>" 4
}')
```

1. Replace **<path_to_pull_secret_file>** with the path to the local file containing the downloaded pull secret from Red Hat OpenShift Cluster Manager at console.redhat.com.

2. Replace **<path_to_ssh_pub_key>** with the path to the public SSH key required to access the host. If you do not set this value, you cannot access the host while in discovery mode.

3. Replace **<infraenv_name>** with the plain text name for the **InfraEnv** resource.

4. Replace **<iso_image_type>** with the ISO image type, either **full-iso** or **minimal-iso**.

c. Post the **$INFRA_ENV_REQUEST** to the /v2/infra-envs API and set the **$INFRA_ENV_ID** variable:

```
$ INFRA_ENV_ID=$(curl "$API_URL/api/assisted-install/v2/infra-envs" -H "Authorization:
Bearer ${API_TOKEN}" -H 'accept: application/json' -H 'Content-Type: application/json' -
d "$INFRA_ENV_REQUEST" | tee /dev/stderr | jq -r '.id')
```

5. Get the URL of the discovery ISO for the cluster host by running the following command:

```
$ curl -s "$API_URL/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID" -H "Authorization:
Bearer ${API_TOKEN}" | jq -r '.download_url'
```

Example output

```
https://api.openshift.com/api/assisted-images/images/41b91e72-c33e-42ee-b80f-
b5c5bbf6431a?
arch=x86_64&image_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NTYwMjYz
NzEsInN1YiI6IjQxYjkxZTcyLWMzM2UtNDJlZS1iODBmLWI1YzViYmY2NDMxYSJ9.1EX_VGaM
NejMhrAvVRBS7PDPIQtbOOc8LtG8OukE1a4&type=minimal-iso&version=4.12
```

6. Download the ISO:

```
$ curl -L -s '<iso_url>' --output rhcos-live-minimal.iso 1
```

1 Replace **<iso_url>** with the URL for the ISO from the previous step.

7. Boot the new worker host from the downloaded **rhcos-live-minimal.iso**.

8. Get the list of hosts in the cluster that are *not* installed. Keep running the following command until the new host shows up:

```
$ curl -s "$API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID" -H "Authorization: Bearer
${API_TOKEN}" | jq -r '.hosts[] | select(.status != "installed").id'
```

Example output

```
2294ba03-c264-4f11-ac08-2f1bb2f8c296
```

9. Set the **$HOST_ID** variable for the new host, for example:

```
$ HOST_ID=<host_id> 1
```

1 Replace **<host_id>** with the host ID from the previous step.

10. Check that the host is ready to install by running the following command:

NOTE

Ensure that you copy the entire command including the complete **jq** expression.

```
$ curl -s $API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID -H "Authorization: Bearer
${API_TOKEN}" | jq '
def host_name($host):
```

```
    if (.suggested_hostname // "") == "" then
        if (.inventory // "") == "" then
            "Unknown hostname, please wait"
        else
            .inventory | fromjson | .hostname
        end
    else
        .suggested_hostname
    end;

def is_notable($validation):
    ["failure", "pending", "error"] | any(. == $validation.status);

def notable_validations($validations_info):
    [
        $validations_info // "{}"
        | fromjson
        | to_entries[].value[]
        | select(is_notable(.))
    ];

{
    "Hosts validations": {
        "Hosts": [
            .hosts[]
            | select(.status != "installed")
            | {
                "id": .id,
                "name": host_name(.),
                "status": .status,
                "notable_validations": notable_validations(.validations_info)
            }
        ]
    },
    "Cluster validations info": {
        "notable_validations": notable_validations(.validations_info)
    }
}
' -r
```

Example output

```
{
  "Hosts validations": {
    "Hosts": [
      {
        "id": "97ec378c-3568-460c-bc22-df54534ff08f",
        "name": "localhost.localdomain",
        "status": "insufficient",
        "notable_validations": [
          {
            "id": "ntp-synced",
            "status": "failure",
            "message": "Host couldn't synchronize with any NTP server"
          },
          {
```

```
            "id": "api-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          },
          {
            "id": "api-int-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          },
          {
            "id": "apps-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          }
        ]
      }
    ]
  },
  "Cluster validations info": {
    "notable_validations": []
  }
}
```

11. When the previous command shows that the host is ready, start the installation using the /v2/infra-envs/{infra_env_id}/hosts/{host_id}/actions/install API by running the following command:

```
$ curl -X POST -s "$API_URL/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID/hosts/$HOST_ID/actions/install"  -H "Authorization: Bearer ${API_TOKEN}"
```

12. As the installation proceeds, the installation generates pending certificate signing requests (CSRs) for the host.

IMPORTANT

You must approve the CSRs to complete the installation.

Keep running the following API call to monitor the cluster installation:

```
$ curl -s "$API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID" -H "Authorization: Bearer ${API_TOKEN}" | jq '{
   "Cluster day-2 hosts":
      [
         .hosts[]
         | select(.status != "installed")
         | {id, requested_hostname, status, status_info, progress, status_updated_at, updated_at, infra_env_id, cluster_id, created_at}
      ]
}'
```

Example output

```
{
```

```
  "Cluster day-2 hosts": [
    {
      "id": "a1c52dde-3432-4f59-b2ae-0a530c851480",
      "requested_hostname": "control-plane-1",
      "status": "added-to-existing-cluster",
      "status_info": "Host has rebooted and no further updates will be posted. Please check
console for progress and to possibly approve pending CSRs",
      "progress": {
        "current_stage": "Done",
        "installation_percentage": 100,
        "stage_started_at": "2022-07-08T10:56:20.476Z",
        "stage_updated_at": "2022-07-08T10:56:20.476Z"
      },
      "status_updated_at": "2022-07-08T10:56:20.476Z",
      "updated_at": "2022-07-08T10:57:15.306369Z",
      "infra_env_id": "b74ec0c3-d5b5-4717-a866-5b6854791bd3",
      "cluster_id": "8f721322-419d-4eed-aa5b-61b50ea586ae",
      "created_at": "2022-07-06T22:54:57.161614Z"
    }
  ]
}
```


13. Optional: Run the following command to see all the events for the cluster:

```
$ curl -s "$API_URL/api/assisted-install/v2/events?cluster_id=$CLUSTER_ID" -H
"Authorization: Bearer ${API_TOKEN}" | jq -c '.[] | {severity, message, event_time, host_id}'
```

Example output


```
{"severity":"info","message":"Host compute-0: updated status from insufficient to known (Host
is ready to be installed)","event_time":"2022-07-08T11:21:46.346Z","host_id":"9d7b3b44-
1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from known to installing
(Installation is in progress)","event_time":"2022-07-08T11:28:28.647Z","host_id":"9d7b3b44-
1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from installing to installing-in-
progress (Starting installation)","event_time":"2022-07-
08T11:28:52.068Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Uploaded logs for host compute-0 cluster 8f721322-419d-4eed-
aa5b-61b50ea586ae","event_time":"2022-07-08T11:29:47.802Z","host_id":"9d7b3b44-1125-
4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from installing-in-progress to
added-to-existing-cluster (Host has rebooted and no further updates will be posted. Please
check console for progress and to possibly approve pending CSRs)","event_time":"2022-07-
08T11:29:48.259Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host: compute-0, reached installation stage
Rebooting","event_time":"2022-07-08T11:29:48.261Z","host_id":"9d7b3b44-1125-4ad0-9b14-
76550087b445"}
```


14. Log in to the cluster and approve the pending CSRs to complete the installation.

Verification

- Check that the new host was successfully added to the cluster with a status of **Ready**:

```
$ oc get nodes
```

Example output

```
NAME                         STATUS   ROLES          AGE   VERSION
control-plane-1.example.com  Ready    master,worker  56m   v1.25.0
compute-1.example.com        Ready    worker         11m   v1.25.0
```

12.5. INSTALLING A PRIMARY CONTROL PLANE NODE ON A HEALTHY CLUSTER

This procedure describes how to install a primary control plane node on a healthy OpenShift Container Platform cluster.

If the cluster is unhealthy, additional operations are required before they can be managed. See Installing a primary control plane node on an unhealthy cluster for additional information.

Prerequisites

- You are using OpenShift Container Platform 4.11 or newer with the correct **etcd-operator** version.
- You have installed a healthy cluster with a minimum of three nodes.
- You have assigned **role: master** to a single node.

Procedure

1. Review and approve CSRs
   a. Review the **CertificateSigningRequests** (CSRs):

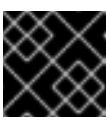
   ```
   $ oc get csr | grep Pending
   ```

   Example output

   ```
   csr-5sd59   8m19s   kubernetes.io/kube-apiserver-client-kubelet
   system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>
   Pending
   csr-xzqts   10s     kubernetes.io/kubelet-serving                 system:node:worker-6
   <none>              Pending
   ```

   b. Approve all pending CSRs:

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

   IMPORTANT

   You must approve the CSRs to complete the installation.

2. Confirm the primary node is in **Ready** status:

```
$ oc get nodes
```

Example output

```
NAME       STATUS   ROLES    AGE     VERSION
master-0   Ready    master   4h42m   v1.24.0+3882f8f
worker-1   Ready    worker   4h29m   v1.24.0+3882f8f
master-2   Ready    master   4h43m   v1.24.0+3882f8f
master-3   Ready    master   4h27m   v1.24.0+3882f8f
worker-4   Ready    worker   4h30m   v1.24.0+3882f8f
master-5   Ready    master   105s    v1.24.0+3882f8f
```

NOTE

The **etcd-operator** requires a **Machine** Custom Resources (CRs) referencing the new node when the cluster runs with a functional Machine API.

3. Link the **Machine** CR with **BareMetalHost** and **Node**:

   a. Create the **BareMetalHost** CR with a unique **.metadata.name** value":

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: custom-master3
  namespace: openshift-machine-api
  annotations:
spec:
  automatedCleaningMode: metadata
  bootMACAddress: 00:00:00:00:00:02
  bootMode: UEFI
  customDeploy:
    method: install_coreos
  externallyProvisioned: true
  online: true
  userData:
    name: master-user-data-managed
    namespace: openshift-machine-api
```

```
$ oc create -f <filename>
```

   b. Apply the **BareMetalHost** CR:

```
$ oc apply -f <filename>
```

   c. Create the **Machine** CR using the unique **.machine.name** value:

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  annotations:
    machine.openshift.io/instance-state: externally provisioned
    metal3.io/BareMetalHost: openshift-machine-api/custom-master3
```

```
  finalizers:
  - machine.machine.openshift.io
  generation: 3
  labels:
    machine.openshift.io/cluster-api-cluster: test-day2-1-6qv96
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
  name: custom-master3
  namespace: openshift-machine-api
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: baremetal.cluster.k8s.io/v1alpha1
      customDeploy:
        method: install_coreos
      hostSelector: {}
      image:
        checksum: ""
        url: ""
      kind: BareMetalMachineProviderSpec
      metadata:
        creationTimestamp: null
      userData:
        name: master-user-data-managed
```

```
$ oc create -f <filename>
```

d. Apply the **Machine** CR:

```
$ oc apply -f <filename>
```

e. Link **BareMetalHost**, **Machine**, and **Node** using the **link-machine-and-node.sh** script:

```
#!/bin/bash

# Credit goes to https://bugzilla.redhat.com/show_bug.cgi?id=1801238.
# This script will link Machine object and Node object. This is needed
# in order to have IP address of the Node present in the status of the Machine.

set -x
set -e

machine="$1"
node="$2"

if [ -z "$machine" -o -z "$node" ]; then
    echo "Usage: $0 MACHINE NODE"
    exit 1
fi

uid=$(echo $node | cut -f1 -d':')
node_name=$(echo $node | cut -f2 -d':')

oc proxy &
```

```
proxy_pid=$!
function kill_proxy {
    kill $proxy_pid
}
trap kill_proxy EXIT SIGINT

HOST_PROXY_API_PATH="http://localhost:8001/apis/metal3.io/v1alpha1/namespaces/openshift-machine-api/baremetalhosts"

function wait_for_json() {
    local name
    local url
    local curl_opts
    local timeout

    local start_time
    local curr_time
    local time_diff

    name="$1"
    url="$2"
    timeout="$3"
    shift 3
    curl_opts="$@"
    echo -n "Waiting for $name to respond"
    start_time=$(date +%s)
    until curl -g -X GET "$url" "${curl_opts[@]}" 2> /dev/null | jq '.' 2> /dev/null > /dev/null; do
        echo -n "."
        curr_time=$(date +%s)
        time_diff=$(($curr_time - $start_time))
        if [[ $time_diff -gt $timeout ]]; then
            echo "\nTimed out waiting for $name"
            return 1
        fi
        sleep 5
    done
    echo " Success!"
    return 0
}
wait_for_json oc_proxy "${HOST_PROXY_API_PATH}" 10 -H "Accept: application/json" -H "Content-Type: application/json"

addresses=$(oc get node -n openshift-machine-api ${node_name} -o json | jq -c '.status.addresses')

machine_data=$(oc get machine -n openshift-machine-api -o json ${machine})
host=$(echo "$machine_data" | jq '.metadata.annotations["metal3.io/BareMetalHost"]' | cut -f2 -d/ | sed 's/"//g')

if [ -z "$host" ]; then
    echo "Machine $machine is not linked to a host yet." 1>&2
    exit 1
fi

# The address structure on the host doesn't match the node, so extract
```

```
# the values we want into separate variables so we can build the patch
# we need.
hostname=$(echo "${addresses}" | jq '.[] | select(. | .type == "Hostname") | .address' | sed
's/"//g')
ipaddr=$(echo "${addresses}" | jq '.[] | select(. | .type == "InternalIP") | .address' | sed
's/"//g')

host_patch='
{
  "status": {
    "hardware": {
      "hostname": "'${hostname}'",
      "nics": [
        {
          "ip": "'${ipaddr}'",
          "mac": "00:00:00:00:00:00",
          "model": "unknown",
          "speedGbps": 10,
          "vlanId": 0,
          "pxe": true,
          "name": "eth1"
        }
      ],
      "systemVendor": {
        "manufacturer": "Red Hat",
        "productName": "product name",
        "serialNumber": ""
      },
      "firmware": {
        "bios": {
          "date": "04/01/2014",
          "vendor": "SeaBIOS",
          "version": "1.11.0-2.el7"
        }
      },
      "ramMebibytes": 0,
      "storage": [],
      "cpu": {
        "arch": "x86_64",
        "model": "Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz",
        "clockMegahertz": 2199.998,
        "count": 4,
        "flags": []
      }
    }
  }
}
'

echo "PATCHING HOST"
echo "${host_patch}" | jq .

curl -s \
    -X PATCH \
    ${HOST_PROXY_API_PATH}/${host}/status \
    -H "Content-type: application/merge-patch+json" \
```

```
    -d "${host_patch}"

oc get baremetalhost -n openshift-machine-api -o yaml "${host}"
```

```
$ bash link-machine-and-node.sh custom-master3 worker-5
```

4. Confirm **etcd** members:

```
$ oc rsh -n openshift-etcd etcd-worker-2
etcdctl member list -w table
```

Example output

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
```

5. Confirm the **etcd-operator** configuration applies to all nodes:

```
$ oc get clusteroperator etcd
```

Example output

```
NAME   VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
etcd   4.11.5    True        False         False      5h54m
```

6. Confirm **etcd-operator** health:

```
$ oc rsh -n openshift-etcd etcd-worker-0
etcdctl endpoint health
```

Example output

```
192.168.111.26 is healthy: committed proposal: took = 11.297561ms
192.168.111.25 is healthy: committed proposal: took = 13.892416ms
192.168.111.28 is healthy: committed proposal: took = 11.870755ms
```

7. Confirm node health:

```
$ oc get Nodes
```

Example output

```
NAME       STATUS   ROLES    AGE     VERSION
master-0   Ready    master   6h20m   v1.24.0+3882f8f
worker-1   Ready    worker   6h7m    v1.24.0+3882f8f
master-2   Ready    master   6h20m   v1.24.0+3882f8f
```

```
master-3   Ready    master   6h4m    v1.24.0+3882f8f
worker-4   Ready    worker   6h7m    v1.24.0+3882f8f
master-5   Ready    master   99m     v1.24.0+3882f8f
```

8. Confirm the **ClusterOperators** health:

```
$ oc get ClusterOperators
```

Example output

```
NAME                                       VERSION AVAILABLE PROGRESSING DEGRADED SINCE
MSG
authentication                             4.11.5  True      False       False    5h57m
baremetal                                  4.11.5  True      False       False    6h19m
cloud-controller-manager                   4.11.5  True      False       False    6h20m
cloud-credential                           4.11.5  True      False       False    6h23m
cluster-autoscaler                         4.11.5  True      False       False    6h18m
config-operator                            4.11.5  True      False       False    6h19m
console                                    4.11.5  True      False       False    6h4m
csi-snapshot-controller                    4.11.5  True      False       False    6h19m
dns                                        4.11.5  True      False       False    6h18m
etcd                                       4.11.5  True      False       False    6h17m
image-registry                             4.11.5  True      False       False    6h7m
ingress                                    4.11.5  True      False       False    6h6m
insights                                   4.11.5  True      False       False    6h12m
kube-apiserver                             4.11.5  True      False       False    6h16m
kube-controller-manager                    4.11.5  True      False       False    6h16m
kube-scheduler                             4.11.5  True      False       False    6h16m
kube-storage-version-migrator              4.11.5  True      False       False    6h19m
machine-api                                4.11.5  True      False       False    6h15m
machine-approver                           4.11.5  True      False       False    6h19m
machine-config                             4.11.5  True      False       False    6h18m
marketplace                                4.11.5  True      False       False    6h18m
monitoring                                 4.11.5  True      False       False    6h4m
network                                    4.11.5  True      False       False    6h20m
node-tuning                                4.11.5  True      False       False    6h18m
openshift-apiserver                        4.11.5  True      False       False    6h8m
openshift-controller-manager               4.11.5  True      False       False    6h7m
openshift-samples                          4.11.5  True      False       False    6h12m
operator-lifecycle-manager                 4.11.5  True      False       False    6h18m
operator-lifecycle-manager-catalog         4.11.5  True      False       False    6h19m
operator-lifecycle-manager-pkgsvr          4.11.5  True      False       False    6h12m
service-ca                                 4.11.5  True      False       False    6h19m
storage                                    4.11.5  True      False       False    6h19m
```

9. Confirm the **ClusterVersion**:

```
$ oc get ClusterVersion
```

Example output

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.5    True        False         5h57m   Cluster version is 4.11.5
```

10. Remove the old control plane node:

    a. Delete the **BareMetalHost** CR:

    ```
    $ oc delete bmh -n openshift-machine-api custom-master3
    ```

    b. Confirm the **Machine** is unhealthy:

    ```
    $ oc get machine -A
    ```

    Example output

    ```
    NAMESPACE               NAME                              PHASE     AGE
    openshift-machine-api   custom-master3                    Running   14h
    openshift-machine-api   test-day2-1-6qv96-master-0        Failed    20h
    openshift-machine-api   test-day2-1-6qv96-master-1        Running   20h
    openshift-machine-api   test-day2-1-6qv96-master-2        Running   20h
    openshift-machine-api   test-day2-1-6qv96-worker-0-8w7vr  Running   19h
    openshift-machine-api   test-day2-1-6qv96-worker-0-rxddj  Running   19h
    ```

    c. Delete the **Machine** CR:

    ```
    $ oc delete machine -n openshift-machine-api   test-day2-1-6qv96-master-0
    machine.machine.openshift.io "test-day2-1-6qv96-master-0" deleted
    ```

    d. Confirm removal of the **Node** CR:

    ```
    $ oc get nodes
    ```

    Example output

    ```
    NAME       STATUS   ROLES    AGE   VERSION
    worker-1   Ready    worker   19h   v1.24.0+3882f8f
    master-2   Ready    master   20h   v1.24.0+3882f8f
    master-3   Ready    master   19h   v1.24.0+3882f8f
    worker-4   Ready    worker   19h   v1.24.0+3882f8f
    master-5   Ready    master   15h   v1.24.0+3882f8f
    ```

11. Check **etcd-operator** logs to confirm status of the **etcd** cluster:

    ```
    $ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf
    ```

    Example output

    ```
    E0927 07:53:10.597523       1 base_controller.go:272] ClusterMemberRemovalController
    reconciliation failed: cannot remove member: 192.168.111.23 because it is reported as
    healthy but it doesn't have a machine nor a node resource
    ```

12. Remove the physical machine to allow **etcd-operator** to reconcile the cluster members:

    ```
    $ oc rsh -n openshift-etcd etcd-worker-2
    etcdctl member list -w table; etcdctl endpoint health
    ```

Example output

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
192.168.111.26 is healthy: committed proposal: took = 10.458132ms
192.168.111.25 is healthy: committed proposal: took = 11.047349ms
192.168.111.28 is healthy: committed proposal: took = 11.414402ms
```

Additional resources

Installing a primary control plane node on an unhealthy cluster

12.6. INSTALLING A PRIMARY CONTROL PLANE NODE ON AN UNHEALTHY CLUSTER

This procedure describes how to install a primary control plane node on an unhealthy OpenShift Container Platform cluster.

Prerequisites

- You are using OpenShift Container Platform 4.11 or newer with the correct **etcd-operator** version.
- You have installed a healthy cluster with a minimum of two nodes.
- You have created the Day 2 control plane.
- You have assigned **role: master** to a single node.

Procedure

1. Confirm initial state of the cluster:

```
$ oc get nodes
```

Example output

```
NAME       STATUS     ROLES    AGE   VERSION
worker-1   Ready      worker   20h   v1.24.0+3882f8f
master-2   NotReady   master   20h   v1.24.0+3882f8f
master-3   Ready      master   20h   v1.24.0+3882f8f
worker-4   Ready      worker   20h   v1.24.0+3882f8f
master-5   Ready      master   15h   v1.24.0+3882f8f
```

2. Confirm the **etcd-operator** detects the cluster as unhealthy:

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf
```

Example output

```
E0927 08:24:23.983733       1 base_controller.go:272] DefragController reconciliation failed:
cluster is unhealthy: 2 of 3 members are available, worker-2 is unhealthy
```

3. Confirm the **etcdctl** members:

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl member list -w table
```

Example output

```
+--------+---------+--------+--------------+--------------+---------+
|   ID   | STATUS  |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+--------------+--------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+--------------+--------------+---------+
```

4. Confirm that **etcdctl** reports an unhealthy member of the cluster:

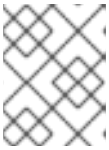
```
$ etcdctl endpoint health
```

Example output

```
{"level":"warn","ts":"2022-09-
27T08:25:35.953Z","logger":"client","caller":"v3/retry_interceptor.go:62","msg":"retrying of
unary invoker failed","target":"etcd-
endpoints://0xc000680380/192.168.111.25","attempt":0,"error":"rpc error: code =
DeadlineExceeded desc = latest balancer error: last connection error: connection error: desc
= \"transport: Error while dialing dial tcp 192.168.111.25: connect: no route to host\""}
192.168.111.28 is healthy: committed proposal: took = 12.465641ms
192.168.111.26 is healthy: committed proposal: took = 12.297059ms
192.168.111.25 is unhealthy: failed to commit proposal: context deadline exceeded
Error: unhealthy cluster
```

5. Remove the unhealthy control plane by deleting the **Machine** Custom Resource:

```
$ oc delete machine -n openshift-machine-api test-day2-1-6qv96-master-2
```



NOTE

The **Machine** and **Node** Custom Resources (CRs) will not be deleted if the unhealthy cluster cannot run successfully.

6. Confirm that **etcd-operator** has not removed the unhealthy machine:

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf -f
```

Example output

```
I0927 08:58:41.249222       1 machinedeletionhooks.go:135] skip removing the deletion hook
from machine test-day2-1-6qv96-master-2 since its member is still present with any of:
[{InternalIP } {InternalIP 192.168.111.26}]
```

7. Remove the unhealthy **etcdctl** member manually:

```
$ oc rsh -n openshift-etcd etcd-worker-3\
etcdctl member list -w table
```

Example output

```
+--------+---------+--------+--------------+--------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+--------------+--------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+--------------+--------------+---------+
```

8. Confirm that **etcdctl** reports an unhealthy member of the cluster:

```
$ etcdctl endpoint health
```

Example output

```
{"level":"warn","ts":"2022-09-
27T10:31:07.227Z","logger":"client","caller":"v3/retry_interceptor.go:62","msg":"retrying of
unary invoker failed","target":"etcd-
endpoints://0xc0000d6e00/192.168.111.25","attempt":0,"error":"rpc error: code =
DeadlineExceeded desc = latest balancer error: last connection error: connection error: desc
= \"transport: Error while dialing dial tcp 192.168.111.25: connect: no route to host\""}
192.168.111.28 is healthy: committed proposal: took = 13.038278ms
192.168.111.26 is healthy: committed proposal: took = 12.950355ms
192.168.111.25 is unhealthy: failed to commit proposal: context deadline exceeded
Error: unhealthy cluster
```

9. Remove the unhealthy cluster by deleting the **etcdctl** member Custom Resource:

```
$ etcdctl member remove 61e2a86084aafa62
```

Example output

```
Member 61e2a86084aafa62 removed from cluster 6881c977b97990d7
```

10. Confirm members of **etcdctl** by running the following command:

```
$ etcdctl member list -w table
```

Example output

```
+----------+---------+--------+--------------+--------------+-------+
|    ID    | STATUS  |  NAME  |  PEER ADDRS  | CLIENT ADDRS |LEARNER|
```

```
+----------+---------+--------+-------------+-------------+-------+
| 2c18942f | started |worker-3|192.168.111.26|192.168.111.26| false |
| ead4f280 | started |worker-5|192.168.111.28|192.168.111.28| false |
+----------+---------+--------+-------------+-------------+-------+
```

11. Review and approve Certificate Signing Requests

    a. Review the Certificate Signing Requests (CSRs):

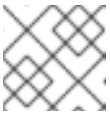
    ```
    $ oc get csr | grep Pending
    ```

    Example output

    ```
    csr-5sd59   8m19s   kubernetes.io/kube-apiserver-client-kubelet
    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   <none>
    Pending
    csr-xzqts   10s     kubernetes.io/kubelet-serving                 system:node:worker-6
    <none>              Pending
    ```

    b. Approve all pending CSRs:

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

    NOTE

    You must approve the CSRs to complete the installation.

12. Confirm ready status of the control plane node:

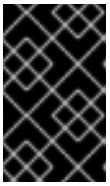
    ```
    $ oc get nodes
    ```

    Example output

    ```
    NAME       STATUS   ROLES    AGE     VERSION
    worker-1   Ready    worker   22h     v1.24.0+3882f8f
    master-3   Ready    master   22h     v1.24.0+3882f8f
    worker-4   Ready    worker   22h     v1.24.0+3882f8f
    master-5   Ready    master   17h     v1.24.0+3882f8f
    master-6   Ready    master   2m52s   v1.24.0+3882f8f
    ```

13. Validate the **Machine**, **Node** and **BareMetalHost** Custom Resources.
    The **etcd-operator** requires **Machine** CRs to be present if the cluster is running with the functional Machine API. **Machine** CRs are displayed during the **Running** phase when present.

14. Create **Machine** Custom Resource linked with **BareMetalHost** and **Node**.
    Make sure there is a **Machine** CR referencing the newly added node.



IMPORTANT

Boot-it-yourself will not create **BareMetalHost** and **Machine** CRs, so you must create them. Failure to create the **BareMetalHost** and **Machine** CRs will generate errors when running **etcd-operator**.

15. Add **BareMetalHost** Custom Resource:

```
$ oc create bmh -n openshift-machine-api custom-master3
```

16. Add **Machine** Custom Resource:

```
$ oc create machine -n openshift-machine-api custom-master3
```

17. Link **BareMetalHost**, **Machine**, and **Node** by running the **link-machine-and-node.sh** script:

```
#!/bin/bash

# Credit goes to https://bugzilla.redhat.com/show_bug.cgi?id=1801238.
# This script will link Machine object and Node object. This is needed
# in order to have IP address of the Node present in the status of the Machine.

set -x
set -e

machine="$1"
node="$2"

if [ -z "$machine" -o -z "$node" ]; then
    echo "Usage: $0 MACHINE NODE"
    exit 1
fi

uid=$(echo $node | cut -f1 -d':')
node_name=$(echo $node | cut -f2 -d':')

oc proxy &
proxy_pid=$!
function kill_proxy {
    kill $proxy_pid
}
trap kill_proxy EXIT SIGINT

HOST_PROXY_API_PATH="http://localhost:8001/apis/metal3.io/v1alpha1/namespaces/openshift-machine-api/baremetalhosts"

function wait_for_json() {
    local name
    local url
    local curl_opts
    local timeout

    local start_time
    local curr_time
    local time_diff
```

```
    name="$1"
    url="$2"
    timeout="$3"
    shift 3
    curl_opts="$@"
    echo -n "Waiting for $name to respond"
    start_time=$(date +%s)
    until curl -g -X GET "$url" "${curl_opts[@]}" 2> /dev/null | jq '.' 2> /dev/null > /dev/null; do
        echo -n "."
        curr_time=$(date +%s)
        time_diff=$(($curr_time - $start_time))
        if [[ $time_diff -gt $timeout ]]; then
            echo "\nTimed out waiting for $name"
            return 1
        fi
        sleep 5
    done
    echo " Success!"
    return 0
}
wait_for_json oc_proxy "${HOST_PROXY_API_PATH}" 10 -H "Accept: application/json" -H
"Content-Type: application/json"

addresses=$(oc get node -n openshift-machine-api ${node_name} -o json | jq -c
'.status.addresses')

machine_data=$(oc get machine -n openshift-machine-api -o json ${machine})
host=$(echo "$machine_data" | jq '.metadata.annotations["metal3.io/BareMetalHost"]' | cut -
f2 -d/ | sed 's/"//g')

if [ -z "$host" ]; then
    echo "Machine $machine is not linked to a host yet." 1>&2
    exit 1
fi

# The address structure on the host doesn't match the node, so extract
# the values we want into separate variables so we can build the patch
# we need.
hostname=$(echo "${addresses}" | jq '.[] | select(. | .type == "Hostname") | .address' | sed
's/"//g')
ipaddr=$(echo "${addresses}" | jq '.[] | select(. | .type == "InternalIP") | .address' | sed 's/"//g')

host_patch='
{
  "status": {
    "hardware": {
      "hostname": "'${hostname}'",
      "nics": [
        {
          "ip": "'${ipaddr}'",
          "mac": "00:00:00:00:00:00",
          "model": "unknown",
          "speedGbps": 10,
          "vlanId": 0,
          "pxe": true,
```

```
        "name": "eth1"
       }
      ],
      "systemVendor": {
       "manufacturer": "Red Hat",
       "productName": "product name",
       "serialNumber": ""
      },
      "firmware": {
       "bios": {
        "date": "04/01/2014",
        "vendor": "SeaBIOS",
        "version": "1.11.0-2.el7"
       }
      },
      "ramMebibytes": 0,
      "storage": [],
      "cpu": {
       "arch": "x86_64",
       "model": "Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz",
       "clockMegahertz": 2199.998,
       "count": 4,
       "flags": []
      }
     }
    }
}
'

echo "PATCHING HOST"
echo "${host_patch}" | jq .

curl -s \
    -X PATCH \
    ${HOST_PROXY_API_PATH}/${host}/status \
    -H "Content-type: application/merge-patch+json" \
    -d "${host_patch}"

oc get baremetalhost -n openshift-machine-api -o yaml "${host}"
```

```
$ bash link-machine-and-node.sh custom-master3 worker-3
```

18. Confirm members of **etcdctl** by running the following command:

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl member list -w table
```

Example output

```
+---------+-------+--------+--------------+--------------+-------+
|   ID    | STATUS|  NAME  |   PEER ADDRS | CLIENT ADDRS |LEARNER|
+---------+-------+--------+--------------+--------------+-------+
| 2c18942f|started|worker-3|192.168.111.26|192.168.111.26| false |
```

```
| ead4f280|started|worker-5|192.168.111.28|192.168.111.28| false |
| 79153c5a|started|worker-6|192.168.111.29|192.168.111.29| false |
+---------+-------+--------+--------------+--------------+-------+
```

19. Confirm the **etcd** operator has configured all nodes:

```
$ oc get clusteroperator etcd
```

Example output

```
NAME   VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
etcd   4.11.5    True        False         False      22h
```

20. Confirm health of **etcdctl**:

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl endpoint health
```

Example output

```
192.168.111.26 is healthy: committed proposal: took = 9.105375ms
192.168.111.28 is healthy: committed proposal: took = 9.15205ms
192.168.111.29 is healthy: committed proposal: took = 10.277577ms
```

21. Confirm the health of the nodes:

```
$ oc get Nodes
```

Example output

```
NAME       STATUS   ROLES    AGE   VERSION
worker-1   Ready    worker   22h   v1.24.0+3882f8f
master-3   Ready    master   22h   v1.24.0+3882f8f
worker-4   Ready    worker   22h   v1.24.0+3882f8f
master-5   Ready    master   18h   v1.24.0+3882f8f
master-6   Ready    master   40m   v1.24.0+3882f8f
```

22. Confirm the health of the **ClusterOperators**:

```
$ oc get ClusterOperators
```

Example output

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
authentication             4.11.5    True        False         False      150m
baremetal                  4.11.5    True        False         False      22h
cloud-controller-manager   4.11.5    True        False         False      22h
cloud-credential           4.11.5    True        False         False      22h
cluster-autoscaler         4.11.5    True        False         False      22h
config-operator            4.11.5    True        False         False      22h
console                    4.11.5    True        False         False      145m
csi-snapshot-controller    4.11.5    True        False         False      22h
```

```
dns                                        4.11.5    True        False         False      22h
etcd                                       4.11.5    True        False         False      22h
image-registry                             4.11.5    True        False         False      22h
ingress                                    4.11.5    True        False         False      22h
insights                                   4.11.5    True        False         False      22h
kube-apiserver                             4.11.5    True        False         False      22h
kube-controller-manager                    4.11.5    True        False         False      22h
kube-scheduler                             4.11.5    True        False         False      22h
kube-storage-version-migrator              4.11.5    True        False         False      148m
machine-api                                4.11.5    True        False         False      22h
machine-approver                           4.11.5    True        False         False      22h
machine-config                             4.11.5    True        False         False      110m
marketplace                                4.11.5    True        False         False      22h
monitoring                                 4.11.5    True        False         False      22h
network                                    4.11.5    True        False         False      22h
node-tuning                                4.11.5    True        False         False      22h
openshift-apiserver                        4.11.5    True        False         False      163m
openshift-controller-manager               4.11.5    True        False         False      22h
openshift-samples                          4.11.5    True        False         False      22h
operator-lifecycle-manager                 4.11.5    True        False         False      22h
operator-lifecycle-manager-catalog         4.11.5    True        False         False      22h
operator-lifecycle-manager-pkgsvr          4.11.5    True        False         False      22h
service-ca                                 4.11.5    True        False         False      22h
storage                                    4.11.5    True        False         False      22h
```

23. Confirm the **ClusterVersion**:

```
$ oc get ClusterVersion
```

Example output

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.5    True        False         22h     Cluster version is 4.11.5
```

Additional resources

Installing a primary control plane node on a healthy cluster

12.7. ADDITIONAL RESOURCES

- Authenticating with the REST API

CHAPTER 13. OPTIONAL: INSTALLING ON NUTANIX

If you install OpenShift Container Platform on Nutanix, the Assisted Installer can integrate the OpenShift Container Platform cluster with the Nutanix platform, which exposes the Machine API to Nutanix and enables autoscaling and dynamically provisioning storage containers with the Nutanix Container Storage Interface (CSI).

13.1. ADDING HOSTS ON NUTANIX WITH THE UI

To add hosts on Nutanix with the user interface (UI), generate the discovery image ISO from the Assisted Installer. Use the minimal discovery image ISO. This is the default setting. The image includes only what is required to boot a host with networking. The majority of the content is downloaded upon boot. The ISO image is about 100MB in size.

Once this is complete, you must create an image for the Nutanix platform and create the Nutanix virtual machines.

Prerequisites

- You have created a cluster profile in the Assisted Installer UI.
- You have a Nutanix cluster environment set up, and made a note of the cluster name and subnet name.

Procedure

1. In **Host discovery**, click the **Add hosts** button and select the installation media.
2. Select **Minimal image file: Provision with virtual media** to download a smaller image that will fetch the data needed to boot.
3. Add an SSH public key so that you can connect to the Nutanix VMs as the **core** user. Having a login to the cluster hosts can provide you with debugging information during the installation.
4. Optional: If the cluster hosts are behind a firewall that requires the use of a proxy, select **Configure cluster-wide proxy settings**. Enter the username, password, IP address and port for the HTTP and HTTPS URLs of the proxy server.
5. Optional: Configure the discovery image if you want to boot it with an ignition file. See Configuring the discovery image for additional details.
6. Click **Generate Discovery ISO**.
7. Copy the **Discovery ISO URL**.
8. In the Nutanix Prism UI, follow the directions to upload the discovery image from the Assisted Installer.
9. In the Nutanix Prism UI, create the control plane (master) VMs through Prism Central.
    a. Enter the **Name**. For example, **control-plane** or **master**.
    b. Enter the **Number of VMs**. This should be 3 for the control plane.
    c. Ensure the remaining settings meet the minimum requirements for control plane hosts.

10. In the Nutanix Prism UI, create the worker VMs through Prism Central.
    a. Enter the **Name**. For example, **worker**.
    b. Enter the **Number of VMs**. You should create at least 2 worker nodes.
    c. Ensure the remaining settings meet the minimum requirements for worker hosts.
11. Return to the Assisted Installer user interface and wait until the Assisted Installer discovers the hosts and each of them have a **Ready** status.
12. Move the **Integrate with your virtualization platform**slider to enable integration with Nutanix.
13. Continue with the installation procedure.

13.2. ADDING HOSTS ON NUTANIX WITH THE API

To add hosts on Nutanix with the API, generate the discovery image ISO from the Assisted Installer. Use the minimal discovery image ISO. This is the default setting. The image includes only what is required to boot a host with networking. The majority of the content is downloaded upon boot. The ISO image is about 100MB in size.

Once this is complete, you must create an image for the Nutanix platform and create the Nutanix virtual machines.

Prerequisites

- You have set up the Assisted Installer API authentication.
- You have created an Assisted Installer cluster profile.
- You have created an Assisted Installer infrastructure environment.
- You have your infrastructure environment ID exported in your shell as **$INFRA_ENV_ID**.
- You have completed the Assisted Installer cluster configuration.
- You have a Nutanix cluster environment set up, and made a note of the cluster name and subnet name.

Procedure

1. Configure the discovery image if you want it to boot with an ignition file.
2. Create a Nutanix cluster configuration file to hold the environment variables:

```
$ touch ~/nutanix-cluster-env.sh
```

```
$ chmod +x ~/nutanix-cluster-env.sh
```

If you have to start a new terminal session, you can reload the environment variables easily. For example:

```
$ source ~/nutanix-cluster-env.sh
```

3. Assign the Nutanix cluster's name to the **NTX_CLUSTER_NAME** environment variable in the configuration file:

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_CLUSTER_NAME=<cluster_name>
EOF
```

   Replace **<cluster_name>** with the name of the Nutanix cluster.

4. Assign the Nutanix cluster's subnet name to the **NTX_SUBNET_NAME** environment variable in the configuration file:

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_SUBNET_NAME=<subnet_name>
EOF
```

   Replace **<subnet_name>** with the name of the Nutanix cluster's subnet.

5. Refresh the API token:

```
$ source refresh-token
```

6. Get the download URL:

```
$ curl -H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID}/downloads/image-url
```

7. Create the Nutanix image configuration file:

```
$ cat << EOF > create-image.json
{
  "spec": {
    "name": "ocp_ai_discovery_image.iso",
    "description": "ocp_ai_discovery_image.iso",
    "resources": {
      "architecture": "X86_64",
      "image_type": "ISO_IMAGE",
      "source_uri": "<image_url>",
      "source_options": {
        "allow_insecure_connection": true
      }
    }
  },
  "metadata": {
    "spec_version": 3,
    "kind": "image"
  }
}
EOF
```

   Replace **<image_url>** with the image URL downloaded from the previous step.

8. Create the Nutanix image:

```
$ curl  -k -u <user>:'<password>' -X 'POST' \
'https://<domain-or-ip>:<port>/api/nutanix/v3/images \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @./create-image.json | jq '.metadata.uuid'
```

Replace **<user>** with the Nutanix user name. Replace **'<password>'** with the Nutanix password. Replace **<domain-or-ip>** with the domain name or IP address of the Nutanix plaform. Replace **<port>** with the port for the Nutanix server. The port defaults to **9440**.

9. Assign the returned UUID to the **NTX_IMAGE_UUID** environment variable in the configuration file:

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_IMAGE_UUID=<uuid>
EOF
```

10. Get the Nutanix cluster UUID:

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/clusters/list' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "kind": "cluster"
}'  | jq '.entities[] | select(.spec.name=="<nutanix_cluster_name>") | .metadata.uuid'
```

Replace **<user>** with the Nutanix user name. Replace **'<password>'** with the Nutanix password. Replace **<domain-or-ip>** with the domain name or IP address of the Nutanix plaform. Replace **<port>** with the port for the Nutanix server. The port defaults to **9440**. Replace **<nutanix_cluster_name>** with the name of the Nutanix cluster.

11. Assign the returned Nutanix cluster UUID to the **NTX_CLUSTER_UUID** environment variable in the configuration file:

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_CLUSTER_UUID=<uuid>
EOF
```

Replace **<uuid>** with the returned UUID of the Nutanix cluster.

12. Get the Nutanix cluster's subnet UUID:

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/subnets/list' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "kind": "subnet",
  "filter": "name==<subnet_name>"
}' | jq '.entities[].metadata.uuid'
```

Replace **<user>** with the Nutanix user name. Replace **'<password>'** with the Nutanix password. Replace **<domain-or-ip>** with the domain name or IP address of the Nutanix plaform. Replace **<port>** with the port for the Nutanix server. The port defaults to **9440**. Replace

<subnet_name> with the name of the cluster's subnet.

13. Assign the returned Nutanix subnet UUID to the **NTX_CLUSTER_UUID** environment variable in the configuration file:

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_SUBNET_UUID=<uuid>
EOF
```

Replace **<uuid>** with the returned UUID of the cluster subnet.

14. Ensure the Nutanix environment variables are set:

```
$ source ~/nutanix-cluster-env.sh
```

15. Create a VM configuration file for each Nutanix host. Create three control plane (master) VMs and at least two worker VMs. For example:

```
$ touch create-master-0.json
```

```
$ cat << EOF > create-master-0.json
{
  "spec": {
    "name": "<host_name>",
    "resources": {
      "power_state": "ON",
      "num_vcpus_per_socket": 1,
      "num_sockets": 16,
      "memory_size_mib": 32768,
      "disk_list": [
        {
          "disk_size_mib": 122880,
          "device_properties": {
            "device_type": "DISK"
          }
        },
        {
          "device_properties": {
            "device_type": "CDROM"
          },
          "data_source_reference": {
            "kind": "image",
            "uuid": "$NTX_IMAGE_UUID"
          }
        }
      ],
      "nic_list": [
        {
          "nic_type": "NORMAL_NIC",
          "is_connected": true,
          "ip_endpoint_list": [
            {
              "ip_type": "DHCP"
            }
          ],
```

```
            "subnet_reference": {
              "kind": "subnet",
              "name": "$NTX_SUBNET_NAME",
              "uuid": "$NTX_SUBNET_UUID"
            }
          }
        ],
        "guest_tools": {
          "nutanix_guest_tools": {
            "state": "ENABLED",
            "iso_mount_state": "MOUNTED"
          }
        }
      },
      "cluster_reference": {
        "kind": "cluster",
        "name": "$NTX_CLUSTER_NAME",
        "uuid": "$NTX_CLUSTER_UUID"
      }
    },
    "api_version": "3.1.0",
    "metadata": {
      "kind": "vm"
    }
}
EOF
```

Replace **<host_name>** with the name of the host.

16. Boot each Nutanix virtual machine:

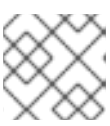
```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/vms' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @./<vm_config_file_name> | jq '.metadata.uuid'
```

Replace **<user>** with the Nutanix user name. Replace **'<password>'** with the Nutanix password. Replace **<domain-or-ip>** with the domain name or IP address of the Nutanix plaform. Replace **<port>** with the port for the Nutanix server. The port defaults to **9440**. Replace **<vm_config_file_name>** with the name of the VM configuration file.

17. Assign the returned VM UUID to a unique environment variable in the configuration file:

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_MASTER_0_UUID=<uuid>
EOF
```

Replace **<uuid>** with the returned UUID of the VM.



NOTE

The environment variable must have a unique name for each VM.

18. Wait until the Assisted Installer has discovered each VM and they have passed validation.

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID"
--header "Content-Type: application/json"
-H "Authorization: Bearer $API_TOKEN"
| jq '.enabled_host_count'
```

19. Modify the cluster definition to enable integration with Nutanix:

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
    "platform_type":"nutanix"
}
' | jq
```

20. Continue with the installation procedure.

13.3. NUTANIX POST-INSTALLATION CONFIGURATION

After installing the OpenShift Container Platform on the Nutanix platform with platform integration enabled, you must update the following Nutanix configuration settings manually:

- **<prismcentral_address>**: The Nutanix Prism Central IP address.
- **<prismcentral_port>**: The Nutanix Prism Central port.
- **<prismelement_address>**: The Nutanix Prism Element IP address.
- **<prismelement_port>**: The Nutanix Prism Element port.
- **<prismelement_clustername>**: The Nutanix Prism Element cluster name.
- **<nutanix_username>**: The Nutanix Prism Element login.
- **<nutanix_password>**: Nutanix Prism Element password.

Prerequisites

- The Assisted Installer has finished installing the cluster successfully.
- The cluster is connected to console.redhat.com.

Procedure

1. Update the Nutanix configuration:

```
$ oc patch infrastructure/cluster --type=merge --patch-file=/dev/stdin <<-EOF
{
  "spec": {
    "platformSpec": {
      "nutanix": {
        "prismCentral": {
```

```
          "address": "<prismcentral_address>", 1
          "port": <prismcentral_port> 2
        },
        "prismElements": [
          {
            "endpoint": {
              "address": "<prismelement_address>", 3
              "port": <prismelement_port> 4
            },
            "name": "<prismelement_clustername>" 5
          }
        ]
      },
      "type": "Nutanix"
    }
  }
}
EOF
```

1. Replace **<prismcentral_address>** with the Nutanix Prism Central IP address.
2. Replace **<prismcentral_port>** with the Nutanix Prism Central port.
3. Replace **<prismelement_address>** with Nutanix Prism Element IP address.
4. Replace **<prismelement_port>** with the Nutanix Prism Element port.
5. Replace **<prismelement_clustername>** with the Nutanix Prism Element cluster name.

For additional details, see Creating a machine set on Nutanix.

2. Update the secret:

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: nutanix-credentials
  namespace: openshift-machine-api
type: Opaque
stringData:
  credentials: |
    [{"type":"basic_auth","data":{"prismCentral":{"username":"
<nutanix_username>","password":"<nutanix_password>"},"prismElements":null}}]
EOF
```

Replace **<nutanix_username>** with the Nutanix Prism Element login. Replace **<nutanix_password>** with the Nutanix Prism Element password.

For additional details, see Configuring the default storage container.

CHAPTER 14. OPTIONAL: INSTALLING ON VSPHERE

If you install OpenShift Container Platform on vSphere, the Assisted Installer can integrate the OpenShift Container Platform cluster with the vSphere platform, which exposes the Machine API to vSphere and enables autoscaling.

14.1. ADDING HOSTS ON VSPHERE

You can add hosts to the Assisted Installer cluster using the online vSphere client or the **govc** vSphere CLI tool. The following procedure demonstrates adding hosts with the **govc** CLI tool. To use the online vSphere Client, refer to the documentation for vSphere.

To add hosts on vSphere with the vSphere **govc** CLI, generate the discovery image ISO from the Assisted Installer. The minimal discovery image ISO is the default setting. This image includes only what is required to boot a host with networking. The majority of the content is downloaded upon boot. The ISO image is about 100MB in size.

Once this is complete, you must create an image for the vSphere platform and create the vSphere virtual machines.

Prerequisites

- You are using vSphere 7.0.2 or higher.
- You have the vSphere **govc** CLI tool installed and configured.
- You have set **clusterSet disk.enableUUID** to **true** in vSphere.
- You have created a cluster in the Assisted Installer UI, or
- You have:
  - Created an Assisted Installer cluster profile and infrastructure environment with the API.
  - Exported your infrastructure environment ID in your shell as **$INFRA_ENV_ID**.
  - Completed the configuration.

Procedure

1. Configure the discovery image if you want it to boot with an ignition file.
2. Generate and download the ISO from the Assisted Installer UI.
   a. In **Host discovery**, click the **Add hosts** button and select the installation media.
   b. Select **Minimal image file: Provision with virtual media** to download a smaller image that will fetch the data needed to boot.
   c. Add an SSH public key so that you can connect to the vSphere VMs as the **core** user. Having a login to the cluster hosts can provide you with debugging information during the installation.
   d. Optional: If the cluster hosts are behind a firewall that requires the use of a proxy, select **Configure cluster-wide proxy settings**. Enter the username, password, IP address and port for the HTTP and HTTPS URLs of the proxy server.

e. Optional: If the cluster hosts are in a network with a re-encrypting man-in-the-middle (MITM) proxy or the cluster needs to trust certificates for other purposes such as container image registries, select **Configure cluster-wide trusted certificates** and add the additional certificates.

f. Optional: Configure the discovery image if you want to boot it with an ignition file. See Configuring the discovery image for additional details.

g. Click **Generate Discovery ISO**.

h. Copy the **Discovery ISO URL**.

i. Download the discovery ISO:

```
$ wget - O vsphere-discovery-image.iso <discovery_url>
```

Replace **<discovery_url>** with the URL from the preceding step.

3. On the command line, power down and destroy any pre-existing virtual machines:

```
$ for VM in $(/usr/local/bin/govc ls /<datacenter>/vm/<folder_name>)
do
  /usr/local/bin/govc vm.power -off $VM
  /usr/local/bin/govc vm.destroy $VM
done
```

Replace **<datacenter>** with the name of the datacenter. Replace **<folder_name>** with the name of the VM inventory folder.

4. Remove pre-existing ISO images from the data store, if there are any:

```
$ govc datastore.rm -ds <iso_datastore> <image>
```

Replace **<iso_datastore>** with the name of the data store. Replace **image** with the name of the ISO image.

5. Upload the Assisted Installer discovery ISO:

```
$ govc datastore.upload -ds <iso_datastore>  vsphere-discovery-image.iso
```

Replace **<iso_datastore>** with the name of the data store.

NOTE

All nodes in the cluster must boot from the discovery image.

6. Boot three control plane (master) nodes:

```
$ govc vm.create -net.adapter <network_adapter_type> \
        -disk.controller <disk_controller_type> \
        -pool=<resource_pool> \
        -c=16 \
        -m=32768 \
        -disk=120GB \
        -disk-datastore=<datastore_file> \
```

```
        -net.address="<nic_mac_address>" \
        -iso-datastore=<iso_datastore> \
        -iso="vsphere-discovery-image.iso" \
        -folder="<inventory_folder>" \
        <hostname>.<cluster_name>.example.com
```

See vm.create for details.

NOTE

The foregoing example illustrates the minimum required resources for control plane nodes.

7. Boot at least two worker nodes:

```
$ govc vm.create -net.adapter <network_adapter_type> \
        -disk.controller <disk_controller_type> \
        -pool=<resource_pool> \
        -c=4 \
        -m=8192 \
        -disk=120GB \
        -disk-datastore=<datastore_file> \
        -net.address="<nic_mac_address>" \
        -iso-datastore=<iso_datastore> \
        -iso="vsphere-discovery-image.iso" \
        -folder="<inventory_folder>" \
        <hostname>.<cluster_name>.example.com
```

See vm.create for details.

NOTE

The foregoing example illustrates the minimum required resources for worker nodes.

8. Ensure the VMs are running:

```
$  govc ls /<datacenter>/vm/<folder_name>
```

Replace **<datacenter>** with the name of the datacenter. Replace **<folder_name>** with the name of the VM inventory folder.

9. After 2 minutes, shut down the VMs:

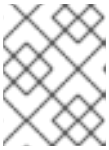
```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
    govc vm.power -s=true $VM
done
```

Replace **<datacenter>** with the name of the datacenter. Replace **<folder_name>** with the name of the VM inventory folder.

10. Set the **disk.enableUUID** setting to **TRUE**:

```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
    govc vm.change -vm $VM -e disk.enableUUID=TRUE
done
```

Replace **<datacenter>** with the name of the datacenter. Replace **<folder_name>** with the name of the VM inventory folder.



NOTE

You must set **disk.enableUUID** to **TRUE** on all of the nodes to enable autoscaling with vSphere.

11. Restart the VMs:

```
$  for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
    govc vm.power -on=true $VM
done
```

Replace **<datacenter>** with the name of the datacenter. Replace **<folder_name>** with the name of the VM inventory folder.

12. Return to the Assisted Installer user interface and wait until the Assisted Installer discovers the hosts and each of them have a **Ready** status.

13. Move the **Integrate with your virtualization platform**slider to enable integration with vSphere.

14. Select roles if needed.

15. In **Networking**, uncheck **Allocate IPs via DHCP server**.

16. Set the API VIP address.

17. Set the Ingress VIP address.

18. Continue with the installation procedure.

14.2. VSPHERE POST-INSTALLATION CONFIGURATION USING THE CLI

After installing an OpenShift Container Platform cluster using the Assisted Installer on vSphere with the platform integration feature enabled, you must update the following vSphere configuration settings manually:

- vCenter username
- vCenter password
- vCenter address
- vCenter cluster
- datacenter

- datastore
- folder

Prerequisites

- The Assisted Installer has finished installing the cluster successfully.
- The cluster is connected to console.redhat.com.

Procedure

1. Generate a base64-encoded username and password for vCenter:

```
$ echo -n "<vcenter_username>" | base64 -w0
```

Replace **<vcenter_username>** with your vCenter username.

```
$ echo -n "<vcenter_password>" | base64 -w0
```

Replace **<vcenter_password>** with your vCenter password.

2. Backup the vSphere credentials:

```
$ oc get secret vsphere-creds -o yaml -n kube-system > creds_backup.yaml
```

3. Edit the vSphere credentials:

```
$ cp creds_backup.yaml vsphere-creds.yaml
```

```
$ vi vsphere-creds.yaml
```

```
apiVersion: v1
data:
  <vcenter_address>.username: <vcenter_username_encoded>
  <vcenter_address>.password: <vcenter_password_encoded>
kind: Secret
metadata:
  annotations:
    cloudcredential.openshift.io/mode: passthrough
  creationTimestamp: "2022-01-25T17:39:50Z"
  name: vsphere-creds
  namespace: kube-system
  resourceVersion: "2437"
  uid: 06971978-e3a5-4741-87f9-2ca3602f2658
type: Opaque
```

Replace **<vcenter_address>** with the vCenter address. Replace **<vcenter_username_encoded>** with the base64-encoded version of your vSphere username. Replace **<vcenter_password_encoded>** with the base64-encoded version of your vSphere password.

4. Replace the vSphere credentials:

```
$ oc replace -f vsphere-creds.yaml
```

5. Redeploy the kube-controller-manager pods:

```
$ oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason":
"recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

6. Backup the vSphere cloud provider configuration:

```
$ oc get cm cloud-provider-config -o yaml -n openshift-config > cloud-provider-
config_backup.yaml
```

7. Edit the cloud provider configuration:

```
$ cloud-provider-config_backup.yaml cloud-provider-config.yaml
```

```
$ vi cloud-provider-config.yaml
```

```
apiVersion: v1
data:
  config: |
    [Global]
    secret-name = "vsphere-creds"
    secret-namespace = "kube-system"
    insecure-flag = "1"

    [Workspace]
    server = "<vcenter_address>"
    datacenter = "<datacenter>"
    default-datastore = "<datastore>"
    folder = "/<datacenter>/vm/<folder>"

    [VirtualCenter "<vcenter_address>"]
    datacenters = "<datacenter>"
kind: ConfigMap
metadata:
  creationTimestamp: "2022-01-25T17:40:49Z"
  name: cloud-provider-config
  namespace: openshift-config
  resourceVersion: "2070"
  uid: 80bb8618-bf25-442b-b023-b31311918507
```

Replace **<vcenter_address>** with the vCenter address. Replace **<datacenter>** with the name of the data center. Replace **<datastore>** with the name of the data store. Replace **<folder>** with the folder containing the cluster VMs.

8. Apply the cloud provider configuration:

```
$ oc apply -f cloud-provider-config.yaml
```

9. Taint the nodes with the **uninitialized** taint:

IMPORTANT

Follow steps 9 through 12 if you are installing OpenShift Container Platform 4.13 or later.

a. Identify the nodes to taint:

```
$ oc get nodes
```

b. Run the following command for each node:

```
$ oc adm taint node <node_name>
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
```

Replace **<node_name>** with the name of the node.

Example

```
$ oc get nodes
NAME              STATUS   ROLES                AGE   VERSION
master-0   Ready    control-plane,master   45h   v1.26.3+379cd9f
master-1   Ready    control-plane,master   45h   v1.26.3+379cd9f
worker-0   Ready    worker                 45h   v1.26.3+379cd9f
worker-1   Ready    worker                 45h   v1.26.3+379cd9f
master-2   Ready    control-plane,master   45h   v1.26.3+379cd9f

$ oc adm taint node master-0
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node master-1
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node master-2
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node worker-0
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node worker-1
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
```

10. Back up the infrastructures configuration:

```
$ oc get infrastructures.config.openshift.io -o yaml >
infrastructures.config.openshift.io.yaml.backup
```

11. Edit the infrastructures configuration:

```
$ cp infrastructures.config.openshift.io.yaml.backup infrastructures.config.openshift.io.yaml
```

```
$ vi infrastructures.config.openshift.io.yaml
```

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: Infrastructure
  metadata:
```

```
    creationTimestamp: "2023-05-07T10:19:55Z"
    generation: 1
    name: cluster
    resourceVersion: "536"
    uid: e8a5742c-6d15-44e6-8a9e-064b26ab347d
  spec:
    cloudConfig:
      key: config
      name: cloud-provider-config
    platformSpec:
      type: VSphere
      vsphere:
        failureDomains:
        - name: assisted-generated-failure-domain
          region: assisted-generated-region
          server: <vcenter_address>
          topology:
            computeCluster: /<data_center>/host/<vcenter_cluster>
            datacenter: <data_center>
            datastore: /<data_center>/datastore/<datastore>
            folder: "/<data_center>/path/to/folder"
            networks:
            - "VM Network"
            resourcePool: /<data_center>/host/<vcenter_cluster>/Resources
          zone: assisted-generated-zone
        nodeNetworking:
          external: {}
          internal: {}
        vcenters:
        - datacenters:
          - <data_center>
          server: <vcenter_address>

kind: List
metadata:
  resourceVersion: ""
```

Replace **<vcenter_address>** with your vCenter address. Replace **<datacenter>** with the name of your vCenter data center. Replace **<datastore>** with the name of your vCenter data store. Replace **<folder>** with the folder containing the cluster VMs. Replace **<vcenter_cluster>** with the vSphere vCenter cluster where OpenShift Container Platform is installed.

12. Apply the infrastructures configuration:

```
$ oc apply -f infrastructures.config.openshift.io.yaml --overwrite=true
```

14.3. VSPHERE POST-INSTALLATION CONFIGURATION USING THE UI

After installing an OpenShift Container Platform cluster using the Assisted Installer on vSphere with the platform integration feature enabled, you must update the following vSphere configuration settings manually:

- vCenter address
- vCenter cluster

- vCenter username
- vCenter password
- Datacenter
- Default data store
- Virtual machine folder

Prerequisites

- The Assisted Installer has finished installing the cluster successfully.
- The cluster is connected to console.redhat.com.

Procedure

1. In the Administrator perspective, navigate to **Home → Overview**.
2. Under **Status**, click **vSphere connection** to open the **vSphere connection configuration** wizard.
3. In the **vCenter** field, enter the network address of the vSphere vCenter server. This can be either a domain name or an IP address. It appears in the vSphere web client URL; for example **https://[your_vCenter_address]/ui**.
4. In the **vCenter cluster** field, enter the name of the vSphere vCenter cluster where OpenShift Container Platform is installed.

IMPORTANT

This step is mandatory if you installed OpenShift Container Platform 4.13 or later.

5. In the **Username** field, enter your vSphere vCenter username.
6. In the **Password** field, enter your vSphere vCenter password.

WARNING

The system stores the username and password in the **vsphere-creds** secret in the **kube-system** namespace of the cluster. An incorrect vCenter username or password makes the cluster nodes unschedulable.

7. In the **Datacenter** field, enter the name of the vSphere data center that contains the virtual machines used to host the cluster; for example, **SDDC-Datacenter**.
8. In the **Default data store** field, enter the vSphere data store that stores the persistent data volumes; for example, **/SDDC-Datacenter/datastore/datastorename**.

WARNING

Updating the vSphere data center or default data store after the configuration has been saved detaches any active vSphere **PersistentVolumes**.

9. In the **Virtual Machine Folder** field, enter the data center folder that contains the virtual machine of the cluster; for example, **/SDDC-Datacenter/vm/ci-ln-hjg4vg2-c61657-t2gzr**. For the OpenShift Container Platform installation to succeed, all virtual machines comprising the cluster must be located in a single data center folder.

10. Click **Save Configuration**. This updates the **cloud-provider-config** file in the **openshift-config** namespace, and starts the configuration process.

11. Reopen the **vSphere connection configuration** wizard and expand the **Monitored operators** panel. Check that the status of the operators is either **Progressing** or **Healthy**.

Verification

The connection configuration process updates operator statuses and control plane nodes. It takes approximately an hour to complete. During the configuration process, the nodes will reboot. Previously bound **PersistentVolumeClaims** objects might become disconnected.

Follow the steps below to monitor the configuration process.

1. Check that the configuration process completed successfully:

   a. In the OpenShift Container Platform Administrator perspective, navigate to **Home → Overview**.

   b. Under **Status** click **Operators**. Wait for all operator statuses to change from **Progressing** to **All succeeded**. A **Failed** status indicates that the configuration failed.

   c. Under **Status**, click **Control Plane**. Wait for the response rate of all Control Pane components to return to 100%. A **Failed** control plane component indicates that the configuration failed.

   A failure indicates that at least one of the connection settings is incorrect. Change the settings in the **vSphere connection configuration** wizard and save the configuration again.

2. Check that you are able to bind **PersistentVolumeClaims** objects by performing the following steps:

   a. Create a **StorageClass** object using the following YAML:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: vsphere-sc
provisioner: kubernetes.io/vsphere-volume
parameters:
 datastore: YOURVCENTERDATASTORE
```

```
 diskformat: thin
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

b. Create a **PersistentVolumeClaims** object using the following YAML:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: test-pvc
 namespace: openshift-config
 annotations:
   volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/vsphere-volume
 finalizers:
   - kubernetes.io/pvc-protection
spec:
 accessModes:
   - ReadWriteOnce
 resources:
   requests:
    storage: 10Gi
 storageClassName: vsphere-sc
 volumeMode: Filesystem
```

For instructions, see Dynamic provisioning in the OpenShift Container Platform documentation. To troubleshoot a **PersistentVolumeClaims** object, navigate to **Storage → PersistentVolumeClaims** in the **Administrator** perspective of the OpenShift Container Platform UI.

CHAPTER 15. TROUBLESHOOTING

There are cases where the Assisted Installer cannot begin the installation or the cluster fails to install properly. In these events, it is helpful to understand the likely failure modes as well as how to troubleshoot the failure.

15.1. PREREQUISITES

- You have created an infrastructure environment using the API or have created a cluster using the UI.

15.2. TROUBLESHOOTING DISCOVERY ISO ISSUES

The Assisted Installer uses an ISO image to run an agent that registers the host to the cluster and performs hardware and network validations before attempting to install OpenShift. You can follow these procedures to troubleshoot problems related to the host discovery.

Once you start the host with the discovery ISO image, the Assisted Installer discovers the host and presents it in the Assisted Service UI.

See Configuring the discovery image for additional details.

15.3. MINIMAL ISO IMAGE

The minimal ISO image should be used when bandwidth over the virtual media connection is limited. It includes only what is required to boot a host with networking. The majority of the content is downloaded upon boot. The resulting ISO image is about 100MB in size compared to 1GB for the full ISO image.

15.3.1. Troubleshooting minimal ISO boot failures

If your environment requires static network configuration to access the Assisted Installer service, any issues with that configuration may prevent the Minimal ISO from booting properly. If the boot screen shows that the host has failed to download the root file system image, verify that any additional network configuration is correct. Switching to a Full ISO image will also allow for easier debugging.

Example rootfs download failure

```
[**    ] A start job is running for Acquire ▮rootfs image (1min 41s / no limit)[
  104.578592] coreos-livepxe-rootfs[922]: curl: (6) Could not resolve host: api.
openshift.com
[  104.579201] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  104.579600] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  104.580107] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   ***] A start job is running for Acquire ▮rootfs image (1min 46s / no limit)[
  109.619825] coreos-livepxe-rootfs[925]: curl: (6) Could not resolve host: api.
openshift.com
[  109.620608] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  109.621053] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  109.621564] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   ***] A start job is running for Acquire ▮rootfs image (1min 51s / no limit)[
  114.647843] coreos-livepxe-rootfs[928]: curl: (6) Could not resolve host: api.
openshift.com
[  114.648464] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  114.648821] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  114.649323] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   ***] A start job is running for Acquire ▮rootfs image (1min 53s / no limit)
```

15.4. VERIFY THE DISCOVERY AGENT IS RUNNING

Prerequisites

- You have created an Infrastructure Environment by using the API or have created a cluster by using the UI.
- You booted a host with the Infrastructure Environment discovery ISO and the host failed to register.
- You have ssh access to the host.
- You provided an SSH public key in the "Add hosts" dialog before generating the Discovery ISO so that you can SSH into your machine without a password.

Procedure

1. Verify that your host machine is powered on.
2. If you selected **DHCP networking**, check that the DHCP server is enabled.
3. If you selected **Static IP, bridges and bonds** networking, check that your configurations are correct.
4. Verify that you can access your host machine using SSH, a console such as the BMC, or a virtual machine console:

   ```
   $ ssh core@<host_ip_address>
   ```

   You can specify private key file using the **-i** parameter if it isn't stored in the default directory.

   ```
   $ ssh -i <ssh_private_key_file> core@<host_ip_address>
   ```

If you fail to ssh to the host, the host failed during boot or it failed to configure the network.

Upon login you should see this message:

Example login

```
**  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  ** **  **  **  **  **
This is a host being installed by the OpenShift Assisted Installer.
It will be installed from scratch during the installation.
The primary service is agent.service. To watch its status, run:
sudo journalctl -u agent.service
To view the agent log, run:
sudo journalctl TAG=agent
**  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  ** **  **  **  **  **
```

If you are not seeing this message it means that the host didn't boot with the assisted-installer ISO. Make sure you configured the boot order properly (The host should boot once from the live-ISO).

5. Check the agent service logs:

```
$ sudo journalctl -u agent.service
```

In the following example, the errors indicate there is a network issue:

Example agent service log screenshot of agent service log

```
Oct 15 11:26:35 localhost systemd[1]: agent. service: Service RestartSec=3s expired, scheduling restart.
Oct 15 11:26:35 localhost systemd[1]: agent. service: Scheduled restart job, restart counter is at 9.
Oct 15 11:26:35 localhost systemd[1]: Stopped agent. service.
Oct 15 11:26:35 localhost systemd[1]: Starting agent. service...
Oct 15 11:26:35 localhost podman[1834]: Trying to pull quay.io/ocpmetal/assisted-installer-agent: lates
Oct 15 11:26:35 localhost podman[1834]:
Get "https://quay.io/v2/": dial top: lookup quay.io on [::1]:53: read udp [::1]:582
Oct 15 11:26:35 localhost podman[1834]: Error: unable to pull quay. io/ocpmetal/assisted-installer-agent:latest: unable to pull
Oct 15 11:26:35 localhost systemd[1]: agent. service: Control process exited, code=exited status=125
Oct 15 11:26:35 localhost systemd[1]: agent. service: Failed with result 'exit-code'.
Oct 15 11:26:35 localhost systemd[1]: Failed to start agent. service.
```

If there is an error pulling the agent image, check the proxy settings. Verify that the host is connected to the network. You can use **nmcli** to get additional information about your network configuration.

15.5. VERIFY THE AGENT CAN ACCESS THE ASSISTED-SERVICE

Prerequisites

- You have created an Infrastructure Environment by using the API or have created a cluster by using the UI.
- You booted a host with the Infrastructure Environment discovery ISO and the host failed to register.
- You verified the discovery agent is running.

Procedure

- Check the agent logs to verify the agent can access the Assisted Service:

```
$ sudo journalctl TAG=agent
```

The errors in the following example indicate that the agent failed to access the Assisted Service.

Example agent log

```
Jul 21 19:39:44 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:39:44" leve
l=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/in
fra-envs/ba747803-f85d-40f4-8af4-01d7f0d8914f/hosts/8daa0b33-d10a-46aa-ab59-ea9be2e0c4d9/instructions?times
tamp=1658432367\": dial tcp: lookup api.stage.openshift.com on 192.168.131.1:53: read udp 192.168.131.11:58
016->192.168.131.1:53: i/o timeout" file="step_processor.go:238" request_id=00a041ba-0314-4d00-83f1-486b36b
d02bb
Jul 21 19:40:44 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:40:44" leve
l=info msg="Query for next steps" file="step_processor.go:223" request_id=6cb39274-7f5b-4099-9894-912702c77
b09
Jul 21 19:40:54 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:40:54" leve
l=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/in
fra-envs/ba747803-f85d-40f4-8af4-01d7f0d8914f/hosts/8daa0b33-d10a-46aa-ab59-ea9be2e0c4d9/instructions?times
tamp=1658432444\": net/http: TLS handshake timeout" file="step_processor.go:238" request_id=6cb39274-7f5b-4
099-9894-912702c77b09
Jul 21 19:41:54 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:41:54" leve
l=info msg="Query for next steps" file="step_processor.go:223" request_id=8ca23a1c-4d5c-494b-8d59-9846fcdff
b9e
```

Check the proxy settings you configured for the cluster. If configured, the proxy must allow access to the Assisted Service URL.

15.6. CORRECTING A HOST'S BOOT ORDER

Once the installation that runs as part of the Discovery Image completes, the Assisted Installer reboots the host. The host must boot from its installation disk to continue forming the cluster. If you have not correctly configured the host's boot order, it will boot from another disk instead, interrupting the installation.

If the host boots the discovery image again, the Assisted Installer will immediately detect this event and set the host's status to **Installing Pending User Action**. Alternatively, if the Assisted Installer does not detect that the host has booted the correct disk within the allotted time, it will also set this host status.

Procedure

- Reboot the host and set its boot order to boot from the installation disk. If you didn't select an installation disk, the Assisted Installer selected one for you. To view the selected installation disk, click to expand the host's information in the host inventory, and check which disk has the "Installation disk" role.

15.7. RECTIFYING PARTIALLY-SUCCESSFUL INSTALLATIONS

There are cases where the Assisted Installer declares an installation to be successful even though it encountered errors:

- If you requested to install OLM operators and one or more failed to install, log into the cluster's console to remediate the failures.
- If you requested to install more than two worker nodes and at least one failed to install, but at least two succeeded, add the failed workers to the installed cluster.