Abstract

Information about the Assisted Installer and its usage
Table of Contents

PREFACE .................................................................................................................. 5

CHAPTER 1. INSTALLING AN ON-PREMISE CLUSTER USING THE ASSISTED INSTALLER ............... 6
  1.1. USING THE ASSISTED INSTALLER .............................................................................. 6
  1.2. API SUPPORT FOR THE ASSISTED INSTALLER .................................................... 7

CHAPTER 2. PREPARING TO INSTALL WITH THE ASSISTED INSTALLER ................................. 8
  2.1. PREREQUISITES .................................................................................................. 8
  2.2. ASSISTED INSTALLER PREREQUISITES .............................................................. 8
    2.2.1. Hardware .................................................................................................... 8
    2.2.2. Networking ................................................................................................. 8
    2.2.3. Preflight validations .................................................................................. 9

CHAPTER 3. INSTALLING WITH THE ASSISTED INSTALLER UI .................................................. 10
  3.1. PRE-INSTALLATION CONSIDERATIONS .................................................................. 10
  3.2. SETTING THE CLUSTER DETAILS ....................................................................... 10
  3.3. OPTIONAL: CONFIGURING STATIC NETWORKS .................................................... 11
  3.4. CONFIGURING OPERATORS .................................................................................. 12
  3.5. ADDING HOSTS TO THE CLUSTER ....................................................................... 12
  3.6. CONFIGURING HOSTS ......................................................................................... 13
  3.7. CONFIGURING NETWORKING ............................................................................. 13
  3.8. PRE-INSTALLATION VALIDATION ....................................................................... 14
  3.9. INSTALLING THE CLUSTER .................................................................................. 15
  3.10. COMPLETING THE INSTALLATION ...................................................................... 15

CHAPTER 4. INSTALLING WITH THE ASSISTED INSTALLER API .................................................. 17
  4.1. OPTIONAL: INSTALLING THE OPENSOURCE CLUSTER MANAGER CLI .................. 17
  4.2. AUTHENTICATING WITH THE REST API ................................................................ 18
  4.3. CONFIGURING THE PULL SECRET ...................................................................... 20
  4.4. REGISTERING A NEW CLUSTER .......................................................................... 21
  4.5. MODIFYING A CLUSTER .................................................................................... 22
  4.6. REGISTERING A NEW INFRASTRUCTURE ENVIRONMENT ................................. 23
  4.7. MODIFYING AN INFRASTRUCTURE ENVIRONMENT ............................................. 24
  4.8. ADDING HOSTS ................................................................................................... 25
  4.9. MODIFYING HOSTS ............................................................................................. 26
  4.10. PRE-INSTALLATION VALIDATION ....................................................................... 27
  4.11. INSTALLING THE CLUSTER ................................................................................ 27

CHAPTER 5. OPTIONAL: ENABLING DISK ENCRYPTION .......................................................... 29
  5.1. ENABLING TPM V2 ENCRYPTION ......................................................................... 29
  5.2. ENABLING TANG ENCRYPTION .......................................................................... 29
  5.3. ADDITIONAL RESOURCES ................................................................................... 30

CHAPTER 6. OPTIONAL: INSTALLING OPERATORS ................................................................... 31
  6.1. INSTALLING OPENSOURCE VIRTUALIZATION ....................................................... 31
  6.2. INSTALLING OPENSOURCE DATA FOUNDATION .................................................. 31

CHAPTER 7. CONFIGURING THE DISCOVERY IMAGE ............................................................ 33
  7.1. CREATING AN IGNITION CONFIGURATION FILE ............................................... 33
  7.2. MODIFYING THE DISCOVERY IMAGE WITH IGNITION ..................................... 34

CHAPTER 8. BOOTING HOSTS WITH THE DISCOVERY IMAGE .................................................... 35
  8.1. CREATING AN ISO IMAGE ON A USB DRIVE ....................................................... 35
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 both x86-64 and arm64 CPU architectures.

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.

- **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. 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\texttt{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

2.2.2. 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 \texttt{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.

**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

### 2.2.3. 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 integrate with vSphere when installing on vSphere

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 RedHat Hybrid Cloud Console.
2. In the menu, click OpenShift.
3. Click Create cluster.
4. Click the Datacenter tab.
5. Under the Assisted Installer section, select Create cluster.
6. Enter a name for the cluster in the Cluster name field.
7. 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.

8. Select the version of OpenShift Container Platform to install.
9. Optional: Select **Install single node Openshift (SNO)** if you want to install OpenShift Container Platform on a single node.

10. 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**

11. Optional: Assisted Installer defaults to using **x86_64** CPU architecture. If you are installing OpenShift Container Platform on **arm64** CPUs, select **Use arm64 CPU architecture**. Keep in mind, some features are not available with **arm64** CPU architecture.

12. 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**

13. 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 address.

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.
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
- OpenShift Data Foundation Logical Volume Manager

Procedure

1. To install OpenShift Virtualization, select **Install OpenShift Virtualization**.
2. To install OpenShift Data Foundation Logical Volume Manager, select **Install OpenShift Data Foundation Logical Volume Manager**.

**IMPORTANT**

OpenShift Data Foundation Logical Volume Manager 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.

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.
   b. Select **Full image file: Provision with physical media** to download the larger full image.
2. 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.
CHAPTER 3. INSTALLING WITH THE ASSISTED INSTALLER UI

3. 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.

4. Configure the discovery image if needed. See **Configuring the discovery image** for additional details.

5. Click **Generate Discovery ISO**.

6. Download the discovery ISO.

7. Boot the host(s) with the discovery image. See **Booting hosts with the discovery image** for additional details.

### 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 configure the hostname, role, and installation disk for each host.

**Procedure**

1. Select a host.

2. From the **Actions** list, select **Change hostname**. You must ensure each host has a valid and unique hostname. If necessary, enter a new name for the host and click **Change**.

3. 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.

4. To the left of the checkbox next to a host name, click to expand the host details. If you have multiple disk drives, you can select a different disk drive to act as the installation disk.

5. Repeat this procedure for each host.

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

### 3.7. 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.

   - **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 telco features.

**Additional resources**
- Network configuration

### 3.8. 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**
3.9. 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.

3.10. 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.
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.
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:

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

Then, generate an API token:

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

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

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

```bash
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:

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

4. Refresh the API token:

```bash
$ source refresh-token
```

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

```bash
```

**Example output**

```json
{
    "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-
```

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 "
   
   { "name": "testcluster", 
   "high_availability_mode": "None", 
   "openshift_version": "4.11", 
   "pull_secret": "$pull_secret[0] | toJSON, 
   "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**

See the `cluster-create-params` model in the API viewer for details on the fields you can set when registering a new cluster. After you create the cluster definition, you can modify the cluster definition and provide values for additional settings.

**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:

   ```bash
   $ 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:

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

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

      ```bash
cat�试 EOF > cluster.json
   ```

   ```json
   {
   "name": "testcluster",
   "openshift_version": "4.11",
   "high_availability_mode": "None",
   ```
3. Assign the returned `cluster_id` to the `CLUSTER_ID` variable and export it:

   ```bash
   $ 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:

   ```bash
   $ 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 ".id"
   ```

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.

### 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.

**Prerequisites**

- You have created a new cluster resource.

**Procedure**

1. Refresh the API token:

   ```bash
   $ source refresh-token
   ```

2. Modify the cluster. For example:
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**

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.

   ```
   $ 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+czCZtVRgB7tqtsMxms2q3TOJZAgfqReKYWm+OLOZTD+DO3H1i1pah/mU3u7uJfT Ug4wEX0Le8zBu9xJVym0BVmSFkzHfJVTn6SIz81NqcalisGWpmkKXVcdnVAX6RsbHlpGKk9YPQarmRCn5KzkeJJK4hrSWpBPjdxfXalp64JBJzew9XkVA3QeXklcFuq7NBuUH9BonrPEmiXNOa41PUP1IWq3mERNgzHzuU8ks/pFuU5HCMmv4qBTOlhi7vidHmHPpqYT/TkuV/i5w0ZZgkkBeLnxWxH0ldrfsFgBYA xnpuTUh8h/4vHg53Gr2mBzXds2ic71mBbbsSrA+zjYNaeYk107UpcCw4jJHspU/rVVDY515 D5gSiuaFPBMcunYPgUx4FMFbfGrtmGLIzTKiLzc0DiSz1jBeTQOX++1nz+KDLBD8CPdi5k4d q7IkapRk85qdEvgaG5RIHMSPSSwDrQ51fD8= user@hostname"' | jq
   ```
a. Optional: You can register a new infrastructure environment by slurping the pull secret file in the request:

```bash
$ 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 \n  --slurpfile pull_secret ~/Downloads/pull-secret.txt \n  --arg cluster_id ${CLUSTER_ID} \
  \n  { \n    "name": "testcluster-infra-env", \n    "image_type": "full-iso", \n    "cluster_id": "$cluster_id", \n    "pull_secret": $pull_secret[0] | tojson \n  } \n  )" | jq ".id"
```

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

```bash
$ 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:

```bash
$ 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:

   ```bash
   $ source refresh-token
   ```

2. Modify the infrastructure environment:

   ```bash
   $ 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.
6. Assign a role to host(s).

**Additional resources**

- Configuring the discovery image
- Booting hosts with the discovery image
- 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> \
      -X PATCH \
      -H "Authorization: Bearer ${API_TOKEN}" \
      -H "Content-Type: application/json" \
      -d ' \
      "host_role":"worker" \
      "host_name" : "worker-1" \
      ' | jq

   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
   ```
CHAPTER 5. OPTIONAL: ENABLING DISK ENCRYPTION

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

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-encryption` 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:

   ```bash
   $ source refresh-token
   
   b. Enable TPM v2 encryption:

   ```bash
   $ 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

```bash
$ source refresh-token
$ 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**.
• 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. 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.

3. 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_encryption.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 OPERATORS

The Assisted Installer can install select operators for you with default configurations. 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.

6.1. INSTALLING OPENSHIFT VIRTUALIZATION

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

- In the operators step of the wizard, enable the "Install OpenShift Virtualization" checkbox.

Additional resources

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

6.2. INSTALLING OPENSHIFT DATA FOUNDATION

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

- Validates that your environment meets the prerequisites outlined below (except for devices being empty, which you should verify).
- Configures the storage to use all available disks.

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 devices you use must be empty, the disks must not include Physical Volumes (PVs), Volume Groups (VGs), or Logical Volumes (LVs) remaining on the disk.

- 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**

- In the "operators" step of the wizard, enable the "Install OpenShift Data Foundation" checkbox.

**Additional resources**

- For more details about OpenShift Data Foundation, see the [OpenShift Documentation](#).
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:

   ```bash
   $ 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:

   ```bash
   $ openssl passwd -6
   
   
   "ignition": { "version": "3.1.0" },
   "passwd": {
     "users": [
       {  
         "name": "core",
         "passwordHash": "$6$spam$M5LGSMGyVD.9XOboxcwsnwNdF4irpJdAWy.1Ry55syyUiUsslzIAHaOrUHr2zg6ruD8YNBPW9kW0H8EnKXyc1"
       }
   ```
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 `SIGNITION_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:
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:

```bash
$ 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/VirtualMedia.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:

```bash
$ 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'
```

Example output

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

```bash
$ 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:

```bash
$ 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.Reboot
```

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

```bash
$ 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.Reboot
```

### 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.
Procedure

1. 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

```sh
#!/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:

```sh
#!/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
```
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" \

3. Modify the **host_role** setting:

   ```sh
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**
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Validation type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connected</td>
<td>non-blocking</td>
<td>Checks that the host has recently communicated with the Assisted Installer.</td>
</tr>
<tr>
<td>has-inventory</td>
<td>non-blocking</td>
<td>Checks that the Assisted Installer received the inventory from the host.</td>
</tr>
<tr>
<td>has-min-cpu-cores</td>
<td>non-blocking</td>
<td>Checks that the number of CPU cores meets the minimum requirements.</td>
</tr>
<tr>
<td>has-min-memory</td>
<td>non-blocking</td>
<td>Checks that the amount of memory meets the minimum requirements.</td>
</tr>
<tr>
<td>has-min-valid-disks</td>
<td>non-blocking</td>
<td>Checks that at least one available disk meets the eligibility criteria.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Validation type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>has-cpu-cores-for-role</td>
<td>blocking</td>
<td>Checks that the number of cores meets the minimum requirements for the host role.</td>
</tr>
<tr>
<td>has-memory-for-role</td>
<td>blocking</td>
<td>Checks that the amount of memory meets the minimum requirements for the host role.</td>
</tr>
<tr>
<td>ignition-downloadable</td>
<td>blocking</td>
<td>For day 2 hosts, checks that the host can download ignition configuration from the day 1 cluster.</td>
</tr>
<tr>
<td>belongs-to-majority-group</td>
<td>blocking</td>
<td>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.</td>
</tr>
<tr>
<td>valid-platform-network-settings</td>
<td>blocking</td>
<td>Checks that the platform is valid for the network settings.</td>
</tr>
<tr>
<td>ntp-synced</td>
<td>non-blocking</td>
<td>Checks if an NTP server has been successfully used to synchronize time on the host.</td>
</tr>
<tr>
<td>container-images-available</td>
<td>non-blocking</td>
<td>Checks if container images have been successfully pulled from the image registry.</td>
</tr>
<tr>
<td>sufficient-installation-disk-speed</td>
<td>blocking</td>
<td>Checks that disk speed metrics from an earlier installation meet requirements, if they exist.</td>
</tr>
<tr>
<td>sufficient-network-latency-requirement-for-role</td>
<td>blocking</td>
<td>Checks that the average network latency between hosts in the cluster meets the requirements.</td>
</tr>
<tr>
<td>sufficient-packet-loss-requirement-for-role</td>
<td>blocking</td>
<td>Checks that the network packet loss between hosts in the cluster meets the requirements.</td>
</tr>
<tr>
<td>has-default-route</td>
<td>blocking</td>
<td>Checks that the host has a default route configured.</td>
</tr>
<tr>
<td>api-domain-name-resolved-correctly</td>
<td>blocking</td>
<td>For a multi node cluster with user managed networking. Checks that the host is able to resolve the API domain name for the cluster.</td>
</tr>
<tr>
<td>api-int-domain-name-resolved-correctly</td>
<td>blocking</td>
<td>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.</td>
</tr>
<tr>
<td>apps-domain-name-resolved-correctly</td>
<td>blocking</td>
<td>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.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Validation type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>compatible-with-cluster-platform</td>
<td>non-blocking</td>
<td>Checks that the host is compatible with the cluster platform</td>
</tr>
<tr>
<td>dns-wildcard-not-configured</td>
<td>blocking</td>
<td>Checks that the wildcard DNS <code>*.&lt;cluster_name&gt;.&lt;base_domain&gt;</code> is not configured, because this causes known problems for OpenShift</td>
</tr>
<tr>
<td>disk-encryption-requirements-satisfied</td>
<td>non-blocking</td>
<td>Checks that the type of host and disk encryption configured meet the requirements.</td>
</tr>
<tr>
<td>non-overlapping-subnets</td>
<td>blocking</td>
<td>Checks that this host does not have any overlapping subnets.</td>
</tr>
<tr>
<td>hostname-unique</td>
<td>blocking</td>
<td>Checks that the hostname is unique in the cluster.</td>
</tr>
<tr>
<td>hostname-valid</td>
<td>blocking</td>
<td>Checks the validity of the hostname, meaning that it matches the general form of hostnames and is not forbidden.</td>
</tr>
<tr>
<td>belongs-to-machine-cidr</td>
<td>blocking</td>
<td>Checks that the host IP is in the address range of the machine CIDR.</td>
</tr>
<tr>
<td>iso-requirements-satisfied</td>
<td>blocking</td>
<td>Validates that the cluster meets the requirements of the Local Storage Operator.</td>
</tr>
<tr>
<td>odf-requirements-satisfied</td>
<td>blocking</td>
<td>Validates that the cluster meets the requirements of the Openshift Data Foundation Operator.</td>
</tr>
</tbody>
</table>

- 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.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Validation type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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:

   ```bash
   $ source refresh-token
   
   2. Get all cluster validations:

   ```bash
   $ 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:

   ```bash
   $ 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Validation type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine-cidr-defined</td>
<td>non-blocking</td>
<td>Checks that the machine network definition exists for the cluster.</td>
</tr>
<tr>
<td>cluster-cidr-defined</td>
<td>non-blocking</td>
<td>Checks that the cluster network definition exists for the cluster.</td>
</tr>
<tr>
<td>service-cidr-defined</td>
<td>non-blocking</td>
<td>Checks that the service network definition exists for the cluster.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Validation type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>no-cidrs-overlapping</td>
<td>blocking</td>
<td>Checks that the defined networks do not overlap.</td>
</tr>
<tr>
<td>networks-same-address-families</td>
<td>blocking</td>
<td>Checks that the defined networks share the same address families (valid address families are IPv4, IPv6)</td>
</tr>
<tr>
<td>network-prefix-valid</td>
<td>blocking</td>
<td>Checks the cluster network prefix to ensure that it is valid and allows enough address space for all hosts.</td>
</tr>
<tr>
<td>machine-cidr-equals-to-calculated-cidr</td>
<td>blocking</td>
<td>For a non user managed networking cluster. Checks that ApiVIP or IngressVIP are members of the machine CIDR if they exist.</td>
</tr>
<tr>
<td>api-vip-defined</td>
<td>non-blocking</td>
<td>For a non user managed networking cluster. Checks that ApiVIP exists.</td>
</tr>
<tr>
<td>api-vip-valid</td>
<td>blocking</td>
<td>For a non user managed networking cluster. Checks if the ApiVIP belongs to the machine CIDR and is not in use.</td>
</tr>
<tr>
<td>ingress-vip-defined</td>
<td>blocking</td>
<td>For a non user managed networking cluster. Checks that IngressVIP exists.</td>
</tr>
<tr>
<td>ingress-vip-valid</td>
<td>non-blocking</td>
<td>For a non user managed networking cluster. Checks if the IngressVIP belongs to the machine CIDR and is not in use.</td>
</tr>
<tr>
<td>all-hosts-are-ready-to-install</td>
<td>blocking</td>
<td>Checks that all hosts in the cluster are in the &quot;ready to install&quot; status.</td>
</tr>
<tr>
<td>sufficient-masters-count</td>
<td>blocking</td>
<td>This validation only applies to multi-node clusters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The cluster must have exactly three masters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the cluster has worker nodes, a minimum of 2 worker nodes must exist.</td>
</tr>
<tr>
<td>dns-domain-defined</td>
<td>non-blocking</td>
<td>Checks that the base DNS domain exists for the cluster.</td>
</tr>
<tr>
<td>pull-secret-set</td>
<td>non-blocking</td>
<td>Checks that the pull secret exists. Does not check that the pull secret is valid or authorized.</td>
</tr>
<tr>
<td>ntp-server-configured</td>
<td>blocking</td>
<td>Checks that each of the host clocks are no more than 4 minutes out of sync with each other.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Validation type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>iso-requirements-satisfied</td>
<td>blocking</td>
<td>Validates that the cluster meets the requirements of the Local Storage Operator.</td>
</tr>
<tr>
<td>odf-requirements-satisfied</td>
<td>blocking</td>
<td>Validates that the cluster meets the requirements of the Openshift Data Foundation Operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The cluster has a minimum of 3 hosts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The cluster has only 3 masters or a minimum of 3 workers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The cluster has 3 eligible disks and each host must have an eligible disk.</td>
</tr>
<tr>
<td>cnv-requirements-satisfied</td>
<td>blocking</td>
<td>Validates that the cluster meets the requirements of Container Native Virtualization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The CPU architecture for the cluster is x86.</td>
</tr>
<tr>
<td>lvm-requirements-satisfied</td>
<td>blocking</td>
<td>Validates that the cluster meets the requirements of the Logical Volume Manager Operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The cluster must be single node.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The cluster must be running Openshift $\geq$ 4.11.0.</td>
</tr>
<tr>
<td>network-type-valid</td>
<td>blocking</td>
<td>Checks the validity of the network type if it exists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The network type must be OpenshiftSDN or OVNKubernetes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- OpenshiftSDN does not support IPv6 or Single Node Openshift.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- OVNKubernetes does not support VIP DHCP allocation.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Type</th>
<th>DNS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterNetwork</td>
<td>DNS</td>
<td>The IP address pools from which Pod IP addresses are allocated.</td>
</tr>
<tr>
<td>serviceNetwork</td>
<td>DNS</td>
<td>The IP address pool for services.</td>
</tr>
<tr>
<td>machineNetwork</td>
<td>DNS</td>
<td>The IP address blocks for machines forming the cluster.</td>
</tr>
<tr>
<td>apiVIP</td>
<td>api.&lt;clusternname.clusterdomain&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>ingressVIP</td>
<td>*.apps.&lt;clusternname.clusterdomain&gt;</td>
<td>The VIP to use for ingress traffic.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Stack</th>
<th>SDN</th>
<th>OVN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dual-stack</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

11.1.1. Limitations
11.1.1. 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:

```yaml
---
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>SNO</th>
<th>Multi-Node Cluster with DHCP mode</th>
<th>Multi-Node Cluster without DHCP mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>clusterNetwork</code></td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>Parameter</td>
<td>SNO</td>
<td>Multi-Node Cluster with DHCP mode</td>
<td>Multi-Node Cluster without DHCP mode</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-----------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>serviceNetwork</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>machineNetwork</td>
<td>Auto-assign possible (*)</td>
<td>Auto-assign possible (*)</td>
<td>Auto-assign possible (*)</td>
</tr>
<tr>
<td>apiVIP</td>
<td>Forbidden</td>
<td>Forbidden</td>
<td>Required</td>
</tr>
<tr>
<td>ingressVIP</td>
<td>Forbidden</td>
<td>Forbidden</td>
<td>Required</td>
</tr>
</tbody>
</table>

(*) 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_vip` and `ingress_vip` 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

```json
---
{
  "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": [
```
11.2.2. Example payload to disable autoallocation

```json
{
    "cidr": "192.168.127.0/24"
}
```

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 addresses.

- 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

```yaml
---
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

```yaml
---
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

```yaml
---
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

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

```bash
$ source refresh-token
```

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

```bash
---
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

```json
---
{
  "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"纽带
    },
    {
      "cidr": "1001:db8::/120"
    }
  ]
}
---
```
11.8.3. Example payload for an OpenShift Container Platform cluster consisting of many nodes

```json
...

{"vip_dhcp_allocation": false,
"network_type": "OVNKubernetes",
"user_managed_networking": false,
"api_vip": "192.168.127.100",
"ingress_vip": "192.168.127.101",
"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 address and the Ingress VIP address must be of the primary IP address family when using dual-stack networking. Currently, Red Hat does not support dual-stack VIPs or dual-stack networking with IPv6 as the primary IP address family. However, Red Hat does support dual-stack networking with IPv4 as the primary IP address family. Therefore, you must place the IPv4 entries before the IPv6 entries.

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. 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.

12.1. PREREQUISITES

- You have created an infrastructure environment using the API or have created a cluster using the UI.

12.2. 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.

12.3. 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.

12.3.1. 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
12.4. 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 podman[1834]: Error: unable to pull quay.io/ocpmetal/assisted-installer-agent:latest; unable to pull 
Oct 15 11:26:35 localhost podman[1834]: agent.service: Control process exited, code=exit status=125 
Oct 15 11:26:35 localhost systemd[1]: agent.service: failed with result 'exit-code'. 
```

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.

**12.5. 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 19:40:44 testinfra-cluster-7c35a054-master-1 next_step.run[000]: time="2021-07-22 19:40:44" level=info msg="Query for next steps" file="step_processor.go:223" request_id=6cb39274-7f5b-4099-9894-912702c77b09
 Jul 21 19:40:54 testinfra-cluster-7c35a054-master-1 next_step.run[000]: time="2021-07-22 19:40:54" level=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/infra-envs/ba747803-f85d-40f4-8af4-01d7f8d8914f/hosts/8daa8b33-d10a-46aa-ab59-e98be2e0c4d9/instructions\timestamp=1659432444\": net/http: TLS handshake timeout" file="step_processor.go:238" request_id=6cb39274-7f5b-4099-9894-912702c77b09
 Jul 21 19:41:54 testinfra-cluster-7c35a054-master-1 next_step.run[000]: time="2021-07-22 19:41:54" level=info msg="Query for next steps" file="step_processor.go:223" request_id=8ca23a1c-4d5c-494b-8d59-9846fcdf28be

Check the proxy settings you configured for the cluster. If configured, the proxy must allow access to the Assisted Service URL.

12.6. 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.

12.7. 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.