Red Hat Advanced Cluster Management for Kubernetes 2.5

Clusters

Read more to learn how to create, import, and manage clusters across cloud providers.

Last Updated: 2023-07-13

Red Hat Advanced Cluster Management for Kubernetes 2.5 Clusters

Read more to learn how to create, import, and manage clusters across cloud providers.

Legal Notice

Abstract

Read more to learn how to create, import, and manage clusters across cloud providers.

Table of Contents

CHAPTER 1. MANAGING YOUR CLUSTERS

Learn how to create, import, and manage clusters across cloud providers by using the Red Hat Advanced Cluster Management for Kubernetes console. Learn how to manage clusters across providers in the following topics:

- Supported providers
- Scaling managed clusters
- Release images
- Creating and modifying bare metal assets
- Creating an infrastructure environment
- Managing credentials overview
- Creating a cluster
- Importing a target managed cluster to the hub cluster
- Creating a cluster in a proxy environment
- Enabling cluster proxy add-ons
- Configuring a specific cluster management role
- Managing cluster labels
- Creating and managing ManagedClusterSets (Technology Preview)
- Using ManagedClusterSets with Placement
- Managing cluster pools (Technology Preview)
- Configuring Ansible Tower tasks to run on managed clusters
- Claiming clusters from cluster pools
- Using hosted control plane clusters (Technology Preview)
- Discovery introduction
- Upgrading your cluster
- Removing a cluster from management
- Cluster backup and restore operator

1.1. CLUSTER LIFECYCLE ARCHITECTURE

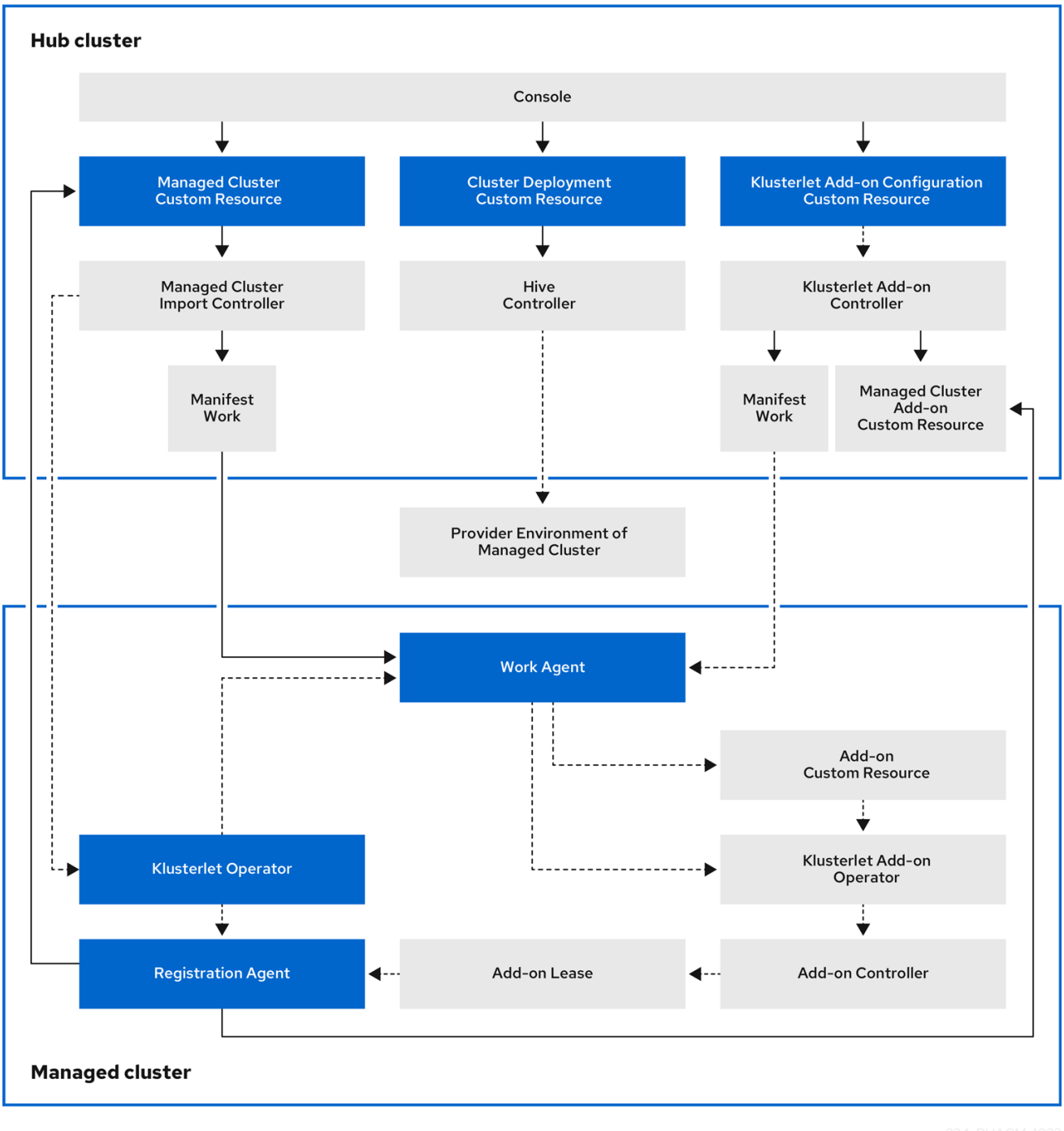
Red Hat Advanced Cluster Management for Kubernetes has two main types of clusters: *hub clusters* and *managed clusters*.

The hub cluster is the main cluster with Red Hat Advanced Cluster Management for Kubernetes installed on it. You can create, manage and monitor other Kubernetes clusters with the hub cluster.

The managed clusters are Kubernetes clusters that are managed by the hub cluster. You can create some clusters by using the Red Hat Advanced Cluster Management hub cluster, while you can also import existing clusters to be managed by the hub cluster.

When you create a managed cluster using Red Hat Advanced Cluster Management, the cluster is created using the Red Hat OpenShift Container Platform cluster installer with the Hive resource. You can find more information about the process of installing clusters with the OpenShift Container Platform installer by reading OpenShift Container Platform installation overview in the OpenShift Container Platform documentation.

The following diagram shows the components that are installed with Red Hat Advanced Cluster Management for cluster management:



The components of the cluster lifecycle management architecture include the following items:

Components on the hub cluster:

- Console: Provides a web-based interface to manage the cluster lifecycle of the Red Hat Advanced Cluster Management managed clusters.
- Hive Controller: Provisions the clusters that you create with Red Hat Advanced Cluster Management. The Hive Controller also detaches and destroys managed clusters that were created by Red Hat Advanced Cluster Management.
- Managed Cluster Import Controller: Deploys the klusterlet operator to the managed clusters.
- Klusterlet Add-on Controller: Deploys the klusterlet add-on operator to managed clusters.

Components on the managed cluster:

- Klusterlet Operator: Deploys the registration and work controllers on the managed cluster.
- Registration Agent: Registers the managed cluster with the hub cluster. The following permissions are automatically created to allow the managed cluster to access the hub cluster:
  - Clusterrole
    - Allows the agent to rotate its certificate
    - Allows the agent to **get**/**list**/**update**/**watch** the clusters that the hub cluster manages
    - Allows the agent to update the status of the clusters that the hub cluster manages
  - Role created in the hub cluster namespace of the hub cluster
    - Allows the managed cluster Registration Agent to **get** or **update** the **coordination.k8s.io** lease
    - Allows the agent to **get**/**list**/**watch** the managed cluster add-on
    - Allows the agent to update the status of managed cluster add-ons
- Work Agent: Applies the manifest work to the managed cluster. The following permission is automatically created to allow the managed cluster to access the hub cluster:
  - Role created in the hub cluster namespace of the hub cluster
    - Allows the Work Agent to send events to the hub cluster
    - Allows the agent to **get**/**list**/**watch**/**update** the **manifestworks** resource
    - Allows the agent to update the status of **manifestworks** resource

1.2. SCALING MANAGED CLUSTERS (TECHNOLOGY PREVIEW)

For clusters that were created by Red Hat Advanced Cluster Management, you can customize and resize your managed cluster specifications, such as virtual machine sizes and number of nodes. To scale managed clusters that were imported from other providers, see Provider managed clusters scaling .

Technology Preview: Many clusters that are managed by Red Hat Advanced Cluster Management for Kubernetes can be scaled by using Red Hat Advanced Cluster Management console or command line, and the **MachinePool** resource.

- Using the **MachinePool** resource is a feature that is not supported for bare metal clusters that were created by Red Hat Advanced Cluster Management.
- A **MachinePool** resource is a Kubernetes resource on the hub cluster that groups the **MachineSet** resources together on the managed cluster.
- The **MachinePool** resource uniformly configures a set of machine resources, including zone configurations, instance type, and root storage.
- With **MachinePool**, you can manually configure the desired number of nodes or configure auto-scaling of nodes on the managed cluster.

1.2.1. Autoscaling

Configuring autoscaling provides the flexibility of your cluster to scale as needed to lower your cost of resources by scaling down when traffic is low, and by scaling up to ensure that there are enough resources when there is a higher demand for resources.

1.2.1.1. Enabling autoscaling

- To enable autoscaling on your **MachinePool** resources using the Red Hat Advanced Cluster Management console, complete the following steps:
  1. In the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Clusters**.
  2. Click the name of your target cluster and select the *Machine pools* tab.
  3. From the machine pools page, select **Enable autoscale** from the *Options* menu for the target machine pool.
  4. Select the minimum and maximum number of machine set replicas. A machine set replica maps directly to a node on the cluster.
     The changes might take several minutes to reflect on the console after you click **Scale**. You can view the status of the scaling operation by clicking **View machines** if the notification of the *Machine pools* tab.
- To enable autoscaling on your **MachinePool** resources using the command line, complete the following steps:
  1. Enter the following command to view your list of machine pools:

     ```
     oc get machinepools -n <managed-cluster-namespace>
     ```

     Replace **managed-cluster-namespace** with the namespace of your target managed cluster.
  2. Enter the following command to edit the YAML file for the machine pool:

     ```
     oc edit machinepool <name-of-MachinePool-resource> -n <namespace-of-managed-cluster>
     ```

     Replace **name-of-MachinePool-resource** with the name of your **MachinePool** resource.

     Replace **namespace-of-managed-cluster** with the name of the namespace of your managed cluster.

3. Delete the **spec.replicas** field from the YAML file.
4. Add the **spec.autoscaling.minReplicas** setting and **spec.autoscaling.maxReplicas** fields to the resource YAML.
5. Add the minimum number of replicas to the **minReplicas** setting.
6. Add the maximum number of replicas into the **maxReplicas** setting.
7. Save the file to submit the changes.

Autoscaling is enabled for the machine pool.

1.2.1.2. Disabling autoscaling

You can disable autoscaling by using the console or the command line.

- To disable autoscaling by using the Red Hat Advanced Cluster Management console, complete the following steps:
  1. In the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Clusters**.
  2. Click the name of your target cluster and select the *Machine pools* tab.
  3. From the machine pools page, select **Disable autoscale** from the *Options* menu for the target machine pool.
  4. Select the number of machine set replicas that you want. A machine set replica maps directly with a node on the cluster.
     It might take several minutes to display in the console after you click **Scale**. You can view the status of the scaling by clicking **View machines** in the notification on the *Machine pools* tab.
- To disable autoscaling by using the command line, complete the following steps:
  1. Enter the following command to view your list of machine pools:

     ```
     oc get machinepools -n <managed-cluster-namespace>
     ```

     Replace **managed-cluster-namespace** with the namespace of your target managed cluster.
  2. Enter the following command to edit the YAML file for the machine pool:

     ```
     oc edit machinepool <name-of-MachinePool-resource> -n <namespace-of-managed-cluster>
     ```

     Replace **name-of-MachinePool-resource** with the name of your **MachinePool** resource.

     Replace **namespace-of-managed-cluster** with the name of the namespace of your managed cluster.
  3. Delete the **spec.autoscaling** field from the YAML file.
  4. Add the **spec.replicas** field to the resource YAML.
  5. Add the number of replicas to the **replicas** setting.

6. Save the file to submit the changes.

Autoscaling is disabled.

1.2.2. Manually scaling your cluster

If you do not want to enable autoscaling of your cluster, you can use the Red Hat Advanced Cluster Management console or the command line to change the static number of replicas that you want your cluster to maintain. This can help to increase or decrease the size, as needed.

- To scale your **MachinePool** resources manually using the Red Hat Advanced Cluster Management console, complete the following steps:

  1. In the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Clusters**.

  2. Click the name of your target cluster and select the *Machine pools* tab.
  Note: If the value in the *Autoscale* field is **Enabled** you must first disable the autoscaling feature by completing the steps in Disabling autoscaling before continuing.

  3. From the *Options* menu for the machine pool, select **Scale machine pool**.

  4. Adjust the number of machine set replicas to scale the machine pool.

- To scale your **MachinePool** resources using the command line, complete the following steps:

  1. Enter the following command to view your list of machine pools:

     ```
     oc get machinepools -n <managed-cluster-namespace>
     ```

     Replace **managed-cluster-namespace** with the namespace of your target managed cluster.

  2. Enter the following command to edit the YAML file for the machine pool:

     ```
     oc edit machinepool <name-of-MachinePool-resource> -n <namespace-of-managed-cluster>
     ```

     Replace **name-of-MachinePool-resource** with the name of your **MachinePool** resource.

     Replace **namespace-of-managed-cluster** with the name of the namespace of your managed cluster.

  3. Update the **spec.replicas** configuration in the YAML to the number of replicas.

  4. Save the file to submit the changes.

Note: Imported managed clusters do not have the same resources as clusters that were created by Red Hat Advanced Cluster Management. For that reason, the procedures for scaling the clusters is different. See the product documentation for your provider, which contains information about how to scale the clusters for imported clusters.

For example, you can see Recommended cluster scaling practices and Manually scaling a MachineSet in the OpenShift Container Platform documentation that applies to the version that you are using.

1.3. RELEASE IMAGES

When you create a cluster on a provider by using Red Hat Advanced Cluster Management for Kubernetes, you must specify a release image to use for the new cluster. The release image specifies which version of Red Hat OpenShift Container Platform is used to build the cluster.

The files that reference the release images are YAML files that are maintained in the **acm-hive-openshift-releases** GitHub repository. Red Hat Advanced Cluster Management uses those files to create the list of the available release images in the console. This includes the latest fast channel images from OpenShift Container Platform. The console only displays the latest release images for the three latest versions of OpenShift Container Platform. For example, you might see the following release images displayed in the console options:

- quay.io/openshift-release-dev/ocp-release:4.6.23-x86_64
- quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64

Note: Only release images with the label of: **visible: 'true'** are available to select when creating clusters in the console. An example of this label in a **ClusterImageSet** resource is provided in the following content:

```
apiVersion: config.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.10.1-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64
```

Additional release images are stored, but are not visible in the console. To view all of the available release images, run **kubectl get clusterimageset** in your CLI. Only the latest versions are in the console to encourage the creation of clusters with the latest release images. In some cases, you might need to create a cluster that is a specific version, which is why the older versions are available. Red Hat Advanced Cluster Management uses those files to create the list of the available release images in the console. This includes the latest fast channel images from OpenShift Container Platform.

The repository contains the **clusterImageSets** directory and the **subscription** directory, which are the directories that you use when working with the release images.

The **clusterImageSets** directory contains the following directories:

- Fast: Contains files that reference the latest versions of the release images for each OpenShift Container Platform version that is supported. The release images in this folder are tested, verified, and supported.
- Releases: Contains files that reference all of the release images for each OpenShift Container Platform version (stable, fast, and candidate channels) **Note:** These releases have not all been tested and determined to be stable.
- Stable: Contains files that reference the latest two stable versions of the release images for each OpenShift Container Platform version that is supported.

Note: By default, the current list of release images is updated one time an hour. After upgrading the product, it may take up to an hour for the list to reflect the recommended release image versions for the new version of the product.

You can curate your own **ClusterImageSets** in three ways:

The first step for any of the three ways is to disable the included subscription that automatically updates the latest fast channel images. The automatic curation of the latest fast **ClusterImageSets** can be disabled by using an installer parameter on the **multiclusterhub** resource. By toggling the **spec.disableUpdateClusterImageSets** parameter between **true** and **false**, the subscription installed with Red Hat Advanced Cluster Management is disabled or enabled, respectively. If you want to curate your own images, set the **spec.disableUpdateClusterImageSets** to **true** to disable the subscription.

Option 1: Specify the image reference for the specific **ClusterImageSet** that you want to use in the console when creating a cluster. Each new entry you specify persists and is available for all future cluster provisions. An example of an entry is: **quay.io/openshift-release-dev/ocp-release:4.6.8-x86_64**.

Option 2: Manually create and apply a **ClusterImageSets** YAML file from the **acm-hive-openshift-releases** GitHub repository.

Option 3: Follow the **README.md** in the **acm-hive-openshift-releases** GitHub repository to enable automatic updates of **ClusterImageSets** from a forked GitHub repository.

The **subscription** directory contains files that specify where the list of release images is pulled from.

The default release images for Red Hat Advanced Cluster Management are provided in a Quay.io directory.

The images are referenced by the files in the acm-hive-openshift-releases GitHub repository for release 2.5.

1.3.1. Creating a release image to deploy a cluster on a different architecture

You can create a cluster on an architecture that is different from the architecture of the hub cluster by manually creating a release image that contains the files for both architectures.

For example, you might need to create an **x86_64** cluster from a hub cluster that is running on the **ppc64le**, **aarch64**, or **s390x** architecture. If you create the release image with both sets of files, the cluster creation succeeds because the new release image enables the OpenShift Container Platform release registry to provide a multi-architecture image manifest.

To create the release image, complete steps similar to the following example for your architecture type:

1. From the OpenShift Container Platform release registry, create a manifest list that includes **x86_64**, **s390x**, **aarch64**, and **ppc64le** release images.

   a. Pull the manifest lists for both architectures in your environment from the Quay repository using the following example commands:

   ```
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-ppc64le
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-s390x
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-aarch64
   ```

   b. Log in to your private repository where you maintain your images:

   ```
   podman login <private-repo>
   ```

   Replace **private-repo** with the path to your repository.

c. Add the release image manifest to your private repository by running the following commands that apply to your environment:

```
podman push quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64 <private-repo>/ocp-release:4.10.1-x86_64
podman push quay.io/openshift-release-dev/ocp-release:4.10.1-ppc64le <private-repo>/ocp-release:4.10.1-ppc64le
podman push quay.io/openshift-release-dev/ocp-release:4.10.1-s390x <private-repo>/ocp-release:4.10.1-s390x
podman push quay.io/openshift-release-dev/ocp-release:4.10.1-aarch64 <private-repo>/ocp-release:4.10.1-aarch64
```

Replace **private-repo** with the path to your repository.

d. Create a manifest for the new information:

```
podman manifest create mymanifest
```

e. Add references to both release images to the manifest list:

```
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-x86_64
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-ppc64le
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-s390x
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-aarch64
```

Replace **private-repo** with the path to your repository.

f. Merge the list in your manifest list with the existing manifest:

```
podman manifest push mymanifest docker://<private-repo>/ocp-release:4.10.1
```

Replace **private-repo** with the path to your repository.

2. On the hub cluster, create a release image that references the manifest in your repository.

a. Create a YAML file that contains information that is similar to the following example:

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: "true"
  name: img4.10.1-appsub
spec:
  releaseImage: <private-repo>/ocp-release:4.10.1
```

Replace **private-repo** with the path to your repository.

b. Run the following command on your hub cluster to apply the changes:

```
oc apply -f <file-name>.yaml
```

Replace **file-name** with the name of the YAML file that you just created.

3. Select the new release image when you create your OpenShift Container Platform cluster.

4. If you deploy the managed cluster using the Red Hat Advanced Cluster Management console, specify the architecture for the managed cluster in the *Architecture* field during the cluster creation process.

The creation process uses the merged release images to create the cluster.

1.3.2. Synchronizing available release images

The release images are updated frequently, so you might want to synchronize the list of release images to ensure that you can select the latest available versions. The release images are available in the acm-hive-openshift-releases GitHub repository for release 2.5.

There are three levels of stability of the release images:

Table 1.1. Stability levels of release images

Category	Description
stable	Fully tested images that are confirmed to install and build clusters correctly.
fast	Partially tested, but likely less stable than a stable version.
candidate	Not tested, but the most current image. Might have some bugs.

Complete the following steps to refresh the list:

1. If the installer-managed **acm-hive-openshift-releases** subscription is enabled, disable the subscription by setting the value of **disableUpdateClusterImageSets** to **true** in the **multiclusterhub** resource.

2. Clone the acm-hive-openshift-releases GitHub repository for release 2.5.

3. Remove the subscription by entering a command that is similar to the following command:

```
oc delete -f subscribe/subscription-fast
```

4. Connect to the stable release images and synchronize your Red Hat Advanced Cluster Management for Kubernetes hub cluster by entering the following command:

```
make subscribe-stable
```

Note: You can only run this **make** command when you are using the Linux or MacOS operating system.

After about one minute, the latest list of **stable** release images is available.

- To synchronize and display the fast release images, enter the following command:

```
make subscribe-fast
```

Note: You can only run this **make** command when you are using the Linux or MacOS operating system.

About one minute after running the command, the list of available **stable** and **fast** release images updates with the currently available images.

- To synchronize and display the **candidate** release images, enter the following command:

```
make subscribe-candidate
```

Note: You can only run this **make** command when you are using the Linux or MacOS operating system.

About one minute after running the command, the list of available **stable**, **fast**, and **candidate** release images updates with the currently available images.

5. View the list of currently available release images in the Red Hat Advanced Cluster Management console when you are creating a cluster.

6. You can unsubscribe from any of these channels to stop viewing the updates by entering a command in the following format:

```
oc delete -f subscribe/subscription-fast
```

1.3.3. Maintaining a custom list of release images when connected

You might want to ensure that you use the same release image for all of your clusters. To simplify, you can create your own custom list of release images that are available when creating a cluster. Complete the following steps to manage your available release images:

1. If the installer-managed **acm-hive-openshift-releases** subscription is enabled, disable it by setting the value of **disableUpdateClusterImageSets** to **true** in the **multiclusterhub** resource.

2. Fork the acm-hive-openshift-releases GitHub repository 2.5 branch .

3. Update the **./subscribe/channel.yaml** file by changing the **spec: pathname** to access your the GitHub name for your forked repository, instead of **stolostron**. This step specifies where the hub cluster retrieves the release images. Your updated content should look similar to the following example:

```
spec:
  type: Git
  pathname: https://github.com/<forked_content>/acm-hive-openshift-releases.git
```

Replace **forked_content** with the path to your forked repository.

4. Add the YAML files for the images that you want available when you create a cluster by using the Red Hat Advanced Cluster Management for Kubernetes console to the *./clusterImageSets/stable/** or *./clusterImageSets/fast/** directory.
Tip: You can retrieve the available YAML files from the main repository by merging changes into your forked repository.

5. Commit and merge your changes to your forked repository.

6. To synchronize your list of fast release images after you have cloned the **acm-hive-openshift-releases** repository, enter the following command to update the fast images:

```
make subscribe-fast
```

Note: You can only run this **make** command when you are using the Linux or MacOS operating system.

After running this command, the list of available fast release images updates with the currently available images in about one minute.

7. By default, only the fast images are listed. To synchronize and display the stable release images, enter the following command:

```
make subscribe-stable
```

Note: You can only run this **make** command when you are using the Linux or MacOS operating system.

After running this command, the list of available stable release images updates with the currently available images in about 1 minute.

8. By default, Red Hat Advanced Cluster Management pre-loads a few ClusterImageSets. You can use the following commands to list what is available and remove the defaults.

```
oc get clusterImageSets
oc delete clusterImageSet <clusterImageSet_NAME>
```

Note: If you have not disabled the installer-managed automatic updates of the **ClusterImageSets** by setting the value of **disableUpdateClusterImageSets** to **true** in the **multiclusterhub** resource, any images that you delete are recreated automatically.

9. View the list of currently available release images in the Red Hat Advanced Cluster Management console when you are creating a cluster.

1.3.4. Maintaining a custom list of release images while disconnected

In some cases, you need to maintain a custom list of release images when the hub cluster has no Internet connection. You can create your own custom list of release images that are available when creating a cluster. Complete the following steps to manage your available release images while disconnected:

1. While you are on a connected system, navigate to the acm-hive-openshift-releases GitHub repository to access the cluster image sets that are available for version 2.5.

2. Copy the **clusterImageSets** directory to a system that can access the disconnected Red Hat Advanced Cluster Management for Kubernetes hub cluster.

3. Add the mapping between the managed cluster and the disconnected repository with your cluster image sets by completing the following steps that fits your managed cluster:

   - For an OpenShift Container Platform managed cluster, see Configuring image registry repository mirroring for information about using your **ImageContentSourcePolicy** object to complete the mapping.

   - For a managed cluster that is not an OpenShift Container Platform cluster, use the **ManageClusterImageRegistry** CRD to override the location of the image sets. See

Importing a cluster with a custom ManagedClusterImageRegistry CRD for information about how to override the cluster for the mapping.

4. Add the YAML files for the images that you want available when you create a cluster by using the Red Hat Advanced Cluster Management console by manually adding the **clusterImageSet** YAML content.

5. Modify the **clusterImageSet** YAML files for the remaining OpenShift Container Platform release images to reference the correct offline repository where you store the images. Your updates should resemble the following example:

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: img4.4.0-rc.6-x86-64
spec:
  releaseImage: IMAGE_REGISTRY_IPADDRESS_or_DNSNAME/REPO_PATH/ocp-release:4.4.0-rc.6-x86_64
```

   Ensure that the images are loaded in the offline image registry that is referenced in the YAML file.

6. Create each of the **clusterImageSets** by entering the following command for each YAML file:

```
oc create -f <clusterImageSet_FILE>
```

   Replace **clusterImageSet_FILE** with the name of the cluster image set file. For example:

```
oc create -f img4.9.9-x86_64.yaml
```

   After running this command for each resource you want to add, the list of available release images will be available.

7. Alternately you can paste the image URL directly in the create cluster console in Red Hat Advanced Cluster Management. Adding the image URL creates new clusterImageSets if they do not exist.

8. View the list of currently available release images in the Red Hat Advanced Cluster Management console when you are creating a cluster.

1.4. CREATING AND MODIFYING BARE METAL ASSETS

Deprecation notice: The procedure for creating bare metal clusters using bare metal assets is deprecated. See Creating a cluster in an on-premises environment for the recommended process.

Bare metal assets are virtual or physical servers that you configure to run your OpenShift Container Platform clusters. Red Hat Advanced Cluster Management for Kubernetes connects to a bare metal asset that your administrator creates. Then you can deploy the bare metal asset in a managed cluster.

The hub cluster inventory controller defines a custom resource definition (CRD) called **BareMetalAsset**, that holds the bare metal asset inventory record. When provisioning a managed cluster, the inventory controller reconciles **BareMetalAsset** inventory records with corresponding **BareMetalHost** resources in the managed cluster.

Red Hat Advanced Cluster Management uses **BareMetalAsset** CRs to provision cluster hardware based

on records entered in a configuration management database (CMDB), or similar system. An external tool or automation polls the CMDB and uses the Red Hat Advanced Cluster Management API to create corresponding **BareMetalAsset** and corresponding **Secret** resources in a hub cluster for subsequent deployment in a managed cluster.

Use the following procedures to create and manage bare metal assets for clusters managed by Red Hat Advanced Cluster Management.

- Prerequisites
- Creating a bare metal asset with the console
- Creating a bare metal asset using the CLI
- Bulk importing bare metal assets with the console
- Modifying a bare metal asset
- Removing a bare metal asset
- Using the REST API to create a bare metal asset

1.4.1. Prerequisites

You need the following prerequisites before creating a bare metal asset:

- A deployed Red Hat Advanced Cluster Management hub cluster on OpenShift Container Platform version 4.6, or later.
- Access for your Red Hat Advanced Cluster Management hub cluster to connect to the bare metal asset.
- A configured bare metal asset, and log in credentials with the required permissions to log in and manage it.
  Note: Credentials for your bare metal asset include the following items for the asset that are provided by your administrator: **user name** password **Baseboard Management Controller (BMC) address** boot NIC MAC address

1.4.2. Creating a bare metal asset with the console

To create a bare metal asset using the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Bare metal assets**. Select **Create bare metal asset**and complete the procedure in the console.

The name for your bare metal asset identifies it when you create a cluster.

The bare metal asset, managed bare metal cluster, and its related secret must be in the same namespace.

+ Users who have access to this namespace can associate this asset to the cluster when creating a cluster.

The Baseboard Management Controller address is the controller that enables communication with the host. The following protocols are supported:

- IPMI, see IPMI 2.0 Specification for more information.

- iDRAC, see Support for Integrated Dell Remote Access Controller 9 (iDRAC9) for more information.
- iRMC, see Data Sheet: FUJITSU Software ServerView Suite integrated Remote Management Controller - iRMC S5 for more information.
- Redfish, see Redfish specification for more information.

The boot NIC MAC address is the MAC address of the network-connected NIC of the host, which is used to provision the host on the bare metal asset.

You can continue with Creating a cluster on bare metal .

1.4.3. Creating a bare metal asset using the CLI

Use a **BareMetalAsset** CR to create a bare metal asset for a particular namespace in the cluster. Each **BareMetalAsset** also has a corresponding **Secret** that contains the Baseboard Management Controller (BMC) credentials and the secret name in the same namespace.

1.4.3.1. Prerequisites

- Install Red Hat Advanced Cluster Management for Kubernetes on a hub cluster.
- Install the Red Hat OpenShift CLI (oc).
- Log in as a user with **cluster-admin** privileges.

1.4.3.2. Create the bare metal asset

1. Install and provision a bare metal asset in your environment.
2. Power on the BMC, and note the IPMI or Redfish BMC address and MAC address for the hardware.
3. Create the following **BareMetalAsset** and **Secret** CR, and save the file as **baremetalasset-cr.yaml**:

```
apiVersion: inventory.open-cluster-management.io/v1alpha1
kind: BareMetalAsset
metadata:
  name: <baremetalasset-machine>
  namespace: <baremetalasset-namespace>
spec:
  bmc:
    address: ipmi://<out_of_band_ip>:<port>
    credentialsName: baremetalasset-machine-secret
  bootMACAddress: "00:1B:44:11:3A:B7"
  hardwareProfile: "hardwareProfile"
  role: "<role>"
  clusterName: "<cluster name>"
---
apiVersion: v1
kind: Secret
metadata:
  name: baremetalasset-machine-secret
type: Opaque
```

```
data:
  username: <username>
  password: <password>
```

- Replace **baremetalasset-machine** with the name of the machine where the bare metal asset is located. When created, the **BareMetalHost** on the managed cluster gets the same name as the corresponding **BareMetalAsset** on the hub cluster. The **BareMetalHost** name should always match the corresponding **BareMetalAsset** name.
- Replace **baremetalasset-namespace** with the cluster namespace where the bare metal asset is created.
- Replace **out_of_band_ip** and **port** with the address and port for your bare metal asset. For Redfish addressing, use the following address format: **redfish://<out-of-band-ip>/redfish/v1/Systems/1**.
- Replace **role** with **worker**, **master**, or leave it empty, depending on the machine role type. The **role** setting is used to match a bare metal asset to specific machine role types in the cluster. **BareMetalAsset** resources with a specified machine role type should not be used to fill another role. The **role** value is used as the value for a label with key **inventory.open-cluster-management.io/role**. This enables a cluster management application or user to query for inventory that is intended for a particular role.
- Replace **cluster_name** with the name of your cluster, which is used by the cluster management application or user to query for inventory that is associated with a particular cluster. Leave this value empty to create the bare metal asset without adding it to a cluster deployment.
- Replace **username** with the username for your secret.
- Replace **password** with the password for your secret.

4. Run the following command to create the **BareMetalAsset** CR:

```
oc create -f baremetalasset-cr.yaml
```

5. Check that the **BareMetalAsset** is created successfully:

```
oc get baremetalassets -A
```

Example output:

```
NAMESPACE         NAME                              AGE
ocp-example-bm    baremetalasset-machine            2m
ocp-example-bm    csv-f24-h27-000-r630-master-1-1   4d21h
```

1.4.4. Bulk importing bare metal assets with the console

You can import bare metal assets in bulk using the Red Hat Advanced Cluster Management for Kubernetes console using a CSV formatted list.

1.4.4.1. Prerequisites

- Install Red Hat Advanced Cluster Management on a hub cluster that manages one or more spoke clusters.

- Install the OpenShift Container Platform CLI (oc).
- Log in as a user with **cluster-admin** privileges.

1.4.4.2. Import the assets

To import a set of bare metal assets, complete the following steps:

1. From the Red Hat Advanced Cluster Management console, select **Cluster management** > **Bare metal assets** in the navigation menu.
2. Select **Import assets**, and import the CSV file that contains the bare metal assets data. The CSV file must have the following header columns:

```
hostName, hostNamespace, bmcAddress, macAddress, role (optional), username, password
```

1.4.5. Modifying a bare metal asset

If you need to modify the settings for a bare metal asset, complete the following steps:

1. In the Red Hat Advanced Cluster Management for Kubernetes console navigation, select: **Infrastructure** > **Bare metal assets**.
2. Select the options menu for the asset that you want to modify in the table.
3. Select **Edit asset**.

1.4.6. Removing a bare metal asset

When a bare metal asset is no longer used for any of the clusters, you can remove it from the list of available bare metal assets. Removing unused assets both simplifies your list of available assets, and prevents the accidental selection of that asset.

To remove a bare metal asset in the console, complete the following steps:

1. In the Red Hat Advanced Cluster Management for Kubernetes console navigation, select: **Infrastructure** > **Bare metal assets**.
2. Select the options menu for the asset that you want to remove in the table.
3. Select **Delete asset**.

1.4.7. Using the REST API to create a bare metal asset

You can use the OpenShift Container Platform REST API to manage bare metal assets for use in your Red Hat Advanced Cluster Management cluster. This is useful when you have a separate CMDB application or database to manage the bare metal assets in your environment.

1.4.7.1. Prerequisites

- Install Red Hat Advanced Cluster Management for Kubernetes on a hub cluster.
- Install the OpenShift Container Platform CLI (oc).
- Log in as a user with **cluster-admin** privileges.

1.4.7.2. Create the bare metal asset

To use the REST API to create a bare metal asset, do the following:

1. Obtain a login token for your hub cluster, and login to the cluster at the command line. For example:

```
oc login --token=<login_token> --server=https://<hub_cluster_api_url>:6443
```

2. Modify the following curl command with the details of the bare metal asset that you want to add to the cluster, and run the command.

```
$ curl --location --request POST '<hub_cluster_api_url>:6443/apis/inventory.open-cluster-management.io/v1alpha1/namespaces/<bare_metal_asset_namespace>/baremetalassets?fieldManager=kubectl-create' \
--header 'Authorization: Bearer <login_token>' \
--header 'Content-Type: application/json' \
--data-raw '{
    "apiVersion": "inventory.open-cluster-management.io/v1alpha1",
    "kind": "BareMetalAsset",
    "metadata": {
        "name": "<baremetalasset_name>",
        "namespace": "<bare_metal_asset_namespace>"
    },
    "spec": {
        "bmc": {
            "address": "ipmi://<ipmi_address>",
            "credentialsName": "<credentials-secret>"
        },
        "bootMACAddress": "<boot_mac_address>",
        "clusterName": "<cluster_name>",
        "hardwareProfile": "hardwareProfile",
        "role": "worker"
    }
}'
```

- Replace **baremetalasset-name** with the name of the bare metal asset. When created, the **BareMetalHost** on the managed cluster gets the same name as the corresponding **BareMetalAsset** on the hub cluster. The **BareMetalHost** name should always match the corresponding **BareMetalAsset** name.

- Replace **baremetalasset-namespace** with the cluster namespace where the bare metal asset is created.

- Replace **out_of_band_ip** and **port** with the address and port for your bare metal asset. For Redfish addressing, use the following address format: **redfish://<out-of-band-ip>/redfish/v1/Systems/1**.

- Replace **role** with **worker**, **master**, or leave it empty, depending on the machine role type. The **role** setting is used to match a bare metal asset to specific machine role types in the cluster. **BareMetalAsset** resources with a specified machine role type should not be used to fill another role. The **role** value is used as the value for a label with key **inventory.open-cluster-management.io/role**. This enables a cluster management application or user to query for inventory that is intended for a particular role.

- Replace **cluster_name** with the name of your cluster, which is used by the cluster

management application or user to query for inventory that is associated with a particular cluster. Leave this value empty to create the bare metal asset without adding it to a cluster deployment.

Note: For the previous curl command, it is assumed that the API server is served over HTTPS and is accessed securely. In a development or test environment, you can pass the **--insecure** parameter.

Tip: You can append **--v=9** to an **oc** command to see the raw output of the resulting action. This can be useful for ascertaining the REST API route for an **oc** command.

1.5. CREATING AN INFRASTRUCTURE ENVIRONMENT

You can use the Red Hat Advanced Cluster Management for Kubernetes console to create an infrastructure environment to manage your hosts and create clusters on those hosts.

- Prerequisites
- Enable Central Infrastructure Management service
  - Manually create the Provisioning custom resource (CR)
  - Enable Central Infrastructure Management on Amazon Web Services
- Creating your infrastructure environment with the console

Infrastructure environments support the following features:

- Zero-touch provisioning of clusters: Deploy clusters using a script. See Deploying distributed units at scale in a disconnected environment in the Red Hat OpenShift Container Platform documentation for more information.
- Late binding: Enable the host to be booted by an infrastructure administrator, and the creator of a cluster can bind a cluster to that host at a later time. The cluster creator does not have to have administrator privileges to the infrastructure when using late binding.
- Dual stack: Deploy clusters that have both IPv4 and IPv6 addresses. Dual stack uses the **OVN-Kubernetes** networking implementation to support multiple subnets.
- Add remote worker nodes: Add remote worker nodes to your clusters after they are created and running, which provides flexibility of adding nodes in other locations for backup purposes.
- Static IP using NMState: Use the NMState API to define static IP addresses for your environment.

1.5.1. Prerequisites

See the following prerequisites before creating an infrastructure environment:

- You must have OpenShift Container Platform deployed on your hub cluster.
- You need Internet access for your Red Hat Advanced Cluster Management hub cluster (connected), or a connection to an internal or mirror registry that has a connection to the Internet (disconnected) to retrieve the required images for creating the environment.

- You need a configured instance of the Central Infrastructure Management (CIM) feature on your hub cluster. See Enabling the Central Infrastructure Management service for the procedure.
- You need an OpenShift Container Platform pull secret. See Using image pull secrets for more information.
- You need your SSH key that is in your **~/.ssh/id_rsa.pub** file, by default.
- You need a configured storage class.
- **Disconnected environment only:** Complete the procedure for Preparing the disconnected environment in the OpenShift Container Platform documentation.

1.5.2. Enabling the Central Infrastructure Management service

The Central Infrastructure Management service is provided with the {mce-short} and deploys OpenShift Container Platform clusters. CIM is deployed when you enable the **MultiClusterHub** Operator on the hub cluster, but must be enabled.

To enable the CIM service, complete the following steps:

Important: Only if your hub cluster is installed on one of the following platforms: bare metal, Red Hat OpenStack Platform, VMware vSphere, or was installed by using the user-provisioned infrastructure (UPI) method and the platform is **None**, complete the following step. Skip this step if your hub cluster is on any other platform.

1. Modify the **Provisioning** resource to allow the Bare Metal Operator to watch all namespaces by running the following command:

```
oc patch provisioning provisioning-configuration --type merge -p '{"spec":
{"watchAllNamespaces": true }}'
```

2. **For disconnected environments:** Create a **ConfigMap** in the *same namespace* as your infrastructure operator to specify the values for **ca-bundle.crt** and **registries.conf** for your mirror registry. Your file **ConfigMap** should resemble the following example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: <mirror-config>
  namespace: "<infrastructure-operator-namespace>"
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    certificate contents
    -----END CERTIFICATE-----

  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
      prefix = ""
      location = "quay.io/edge-infrastructure"
```

```
    mirror-by-digest-only = false

    [[registry.mirror]]
    location = "mirror1.registry.corp.com:5000/edge-infrastructure"
```

1.5.2.1. Creating the *AgentServiceConfig* custom resource

Create the **AgentServiceConfig** custom resource by completing the following steps:

1. **For disconnected environments only:** Save the following **YAML** content in the **agent_service_config.yaml** file and replace the values as needed:

```
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
 name: agent
spec:
  databaseStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <db_volume_size>
  filesystemStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <fs_volume_size>
  mirrorRegistryRef:
    name: <mirror_config>
  unauthenticatedRegistries:
    - <unauthenticated_registry>
  imageStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <img_volume_size>
  osImages:
    - openshiftVersion: "<ocp_version>"
      version: "<ocp_release_version>"
      url: "<iso_url>"
      rootFSUrl: "<root_fs_url>"
      cpuArchitecture: "x86_64"
```

   Replace **mirror_config** with the name of the **ConfigMap** that contains your mirror registry configuration details.

   Include the optional **unauthenticated_registry** parameter if you are using a mirror registry that does not require authentication. Entries on this list are not validated or required to have an entry in the pull secret.

2. **For connected environments only:** Save the following **YAML** content in the **agent_service_config.yaml** file:

```
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
 name: agent
spec:
  databaseStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <db_volume_size>
  filesystemStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <fs_volume_size>
  imageStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <img_volume_size>
```

Replace **db_volume_size** with the volume size for the **databaseStorage** field, for example **10G**. This value specifies how much storage is is allocated for storing files such as database tables and database views for the clusters. You might need to use a higher value if there are many clusters.

Replace **fs_volume_size** with the size of the volume for the **filesystemStorage** field, for example **200M** per cluster and **2-3G** per supported OpenShift Container Platform version. The minimum value that is required is **100G**. This value specifies how much storage is allocated for storing logs, manifests, and **kubeconfig** files for the clusters. You might need to use a higher value if there are many clusters.

Replace **img_volume_size** with the size of the volume for the **imageStorage** field, for example **2G** per operating system image. The minimum size is **50G**. This value specifies how much storage is allocated for the images of the clusters. You need to allow 1 GB of image storage for each instance of Red Hat Enterprise Linux CoreOS that is running. You might need to use a higher value if there are many clusters and instances of Red Hat Enterprise Linux CoreOS.

Replace **ocp_version** with the OpenShift Container Platform version to install, for example, **4.9**.

Replace **ocp_release_version** with the specific install version, for example, **49.83.202103251640-0**.

Replace **iso_url** with the ISO url, for example, **https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/rhcos-4.10.3-x86_64-live.x86_64.iso**. You can find other values at: https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/.

Replace **root_fs_url** with the root FS image URL, for example, **https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/rhcos-4.10.3-x86_64-live-rootfs.x86_64.img**. You can find other values at: https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/.

3. Create the **AgentServiceConfig** custom resource by running the following command:

```
oc create -f agent_service_config.yaml
```

The output might resemble the following example:

```
agentserviceconfig.agent-install.openshift.io/agent created
```

You can verify that it is healthy by checking the **assisted-service** and **assisted-image-service** deployments and ensuring that their pods are ready and running. Continue with Creating your infrastructure environment with the console.

1.5.2.2. Manually create the Provisioning custom resource (CR)

Manually create a **Provisioning** CR to enable services for automated provisioning by using the following command:

```
oc create -f provisioning-configuration.yaml
```

Your CR might resemble the following sample:

```
apiVersion: metal3.io/v1alpha1
kind: Provisioning
metadata:
  name: provisioning-configuration
spec:
  provisioningNetwork: Disabled
  watchAllNamespaces: true
```

1.5.2.3. Enabling Central Infrastructure Management on Amazon Web Services

If you are running your hub cluster on Amazon Web Services and want to enable the CIM service, complete the following additional steps after Enabling CIM:

1. Make sure you are logged in at the hub and find the unique domain configured on the **assisted-image-service** by running the following command:

```
oc get routes --all-namespaces | grep assisted-image-service
```

Your domain might resemble the following example: **assisted-image-service-multicluster-engine.apps.<yourdomain>.com**

2. Make sure you are logged in at the hub and create a new **IngressController** with a unique domain using the **NLB type** parameter. See the following example:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: ingress-controller-with-nlb
  namespace: openshift-ingress-operator
spec:
  domain: nlb-apps.<domain>.com
  routeSelector:
```

```
      matchLabels:
        router-type: nlb
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: External
      providerParameters:
        type: AWS
        aws:
          type: NLB
```

3. Add **<yourdomain>** to the **domain** parameter in **IngressController** by replacing **<domain>** in **nlb-apps.<domain>.com** with **<yourdomain>**.

4. Apply the new **IngressController** by using the following command:

```
oc apply -f ingresscontroller.yaml
```

5. Run the following command to edit the **assisted-image-service** route to use the **nlb-apps** location:

```
oc edit route assisted-image-service -n <namespace>
```

Tip: The default namespace is where you installed the :mce:.

6. Add the following lines to the **assisted-image-service** route:

```
metadata:
  labels:
    router-type: nlb
  name: assisted-image-service
```

7. In the **assisted-image-service** route, find the URL value of **spec.host**. The URL might resemble the following example:
assisted-image-service-multicluster-engine.apps.<yourdomain>.com

8. Replace **apps** in the URL with **nlb-apps** to match the domain configured in the new **IngressController**.

To verify that the CIM service is enabled on Amazon Web Services, complete the following steps:

1. Run the following command to verify that the pods are healthy:

```
oc get pods -n multicluster-engine | grep assist
```

2. Create a new infrastructure environment and ensure that the download URL uses the new **nlb-apps** URL.

1.5.3. Creating your infrastructure environment with the console

To create an infrastructure environment from the Red Hat Advanced Cluster Management console, complete the following steps:

1. From the navigation menu, navigate to **Infrastructure** > **Infrastructure environments** and click **Create infrastructure environment**.

2. Add the following information to your infrastructure environment settings:

   - Name: A unique name for your environment.

   - Network type: Specifies which types of hosts can be added to your environment. You can only use the static IP option when you are using bare metal hosts.

   - Location: Specifies the geographic location of the host. The geographic location can be used to easily determine where your data on a cluster is stored when you are creating the cluster.

   - Labels: Optional field where you can add labels to the infrastructure environment so you can more easily find and group the environment with other environments that share a characteristic. The selections that you made for the network type and location are automatically added to the list of labels.

   - Pull secret: Your OpenShift Container Platform pull secret that enables you to access the OpenShift Container Platform resources.

   - SSH public key: The SSH key that enables the secure communication with the hosts. This is generally in your ~/**.ssh/id_rsa.pub** file, by default.

   - If you want to enable proxy settings across all of your clusters, select the setting to enable it. This requires that you enter the following information:

     - HTTP Proxy URL: The URL that should be used when accessing the discovery service.

     - HTTPS Proxy URL: The secure proxy URL that should be used when accessing the discovery service. Note that the format must be **http**, as **https** is not yet supported.

     - No Proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk ***** to bypass the proxy for all destinations.

You can now continue by adding hosts to your infrastructure environment.

To access an infrastructure environment, select **Infrastructure** > **Host inventory** in the console. Select your infrastructure environment from the list to view the details and hosts for that infrastructure environment.

1.5.4. Adding hosts to an infrastructure environment

You can use the Red Hat Advanced Cluster Management for Kubernetes console to add hosts to an infrastructure environment. Adding the hosts makes it easier to select the already-configured hosts when you are creating a cluster.

Complete the following steps to add a host:

1. From the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Infrastructure environments**.

2. Select the infrastructure environment where you want to add the host to view its settings.

3. Select the *Hosts* tab to view the hosts that are already added to that environment and to add a host. Available hosts might take a few minutes to appear in the table.

4. Select **Discovery ISO** or **Baseboard Management Controller (BMC)** to enter the information for your host.

5. If you select the **Discovery ISO** option, complete the following steps:

    a. Copy the command that is provided in the console to download the ISO or select **Download Discovery ISO**.

    b. Run the command on a bootable device to start each host.

    c. For added security, select **Approve host** for each of the discovered hosts. This additional step offers some protection in case your ISO file is changed and run by an unauthorized person.

    d. Rename the hosts that are named, **localhost** to unique names.

6. If you select **Baseboard Management Controller (BMC)** option, complete the following steps:
Note: The BMC option for adding hosts can only be used when the platform of your Red Hat Advanced Cluster Management hub cluster is bare metal, Red Hat OpenStack Platform, VMware vSphere, or was installed using the user-provisioned infrastructure (UPI) method and the platform is **None**.

    a. Add the connection details for the BMC of your host.

    b. Select **Add host** to start the boot process. The host is automatically booted by using the discovery ISO image, and is added to the list of hosts when it is started.
    When you add a host by using the BMC option, the host is automatically approved.

You can now create an on-premises cluster on this infrastructure environment. See Creating a cluster in an on-premises environment for more information about creating a cluster.

1.6. CREATING A CLUSTER

Learn how to create Red Hat OpenShift Container Platform clusters across cloud providers with Red Hat Advanced Cluster Management for Kubernetes.

Multicluster-engine uses the Hive operator that is provided with OpenShift Container Platform to provision clusters for all providers except the on-premises clusters and hosted control planes. When provisioning the on-premises clusters, multicluster-engine uses the Central Infrastructure Management (CIM) and Assisted Installer function that are provided with OpenShift Container Platform. The hosted clusters for hosted control planes are provisioned by using the HyperShift operator.

- Configuring additional manifests during cluster creation
- Creating a cluster on Amazon Web Services
- Creating a cluster on Microsoft Azure
- Creating a cluster on Google Cloud Platform
- Creating a cluster on VMware vSphere
- Creating a cluster on Red Hat OpenStack Platform

- Creating a cluster on Red Hat Virtualization
- Creating a cluster on bare metal
- Creating a cluster in an on-premises environment

1.6.1. Configuring additional manifests during cluster creation

You can configure additional Kubernetes resource manifests during the installation process of creating your cluster. This can help if you need to configure additional manifests for scenarios such as configuring networking or setting up a load balancer.

Before you create your cluster, you need to add a reference to the **ClusterDeployment** resource that specifies a **ConfigMap** that contains the additional resource manifests.

Note: The **ClusterDeployment** resource and the **ConfigMap** must be in the same namespace. The following examples show how your content might look.

- ConfigMap with resource manifests
  ConfigMap that contains a manifest with another **ConfigMap** resource. The resource manifest **ConfigMap** can contain multiple keys with resource configurations added in a **data.<resource_name>\.yaml** pattern.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: <my-baremetal-cluster-install-manifests>
  namespace: <mynamespace>
data:
  99_metal3-config.yaml: |
    kind: ConfigMap
    apiVersion: v1
    metadata:
      name: metal3-config
      namespace: openshift-machine-api
    data:
      http_port: "6180"
      provisioning_interface: "enp1s0"
      provisioning_ip: "172.00.0.3/24"
      dhcp_range: "172.00.0.10,172.00.0.100"
      deploy_kernel_url: "http://172.00.0.3:6180/images/ironic-python-agent.kernel"
      deploy_ramdisk_url: "http://172.00.0.3:6180/images/ironic-python-agent.initramfs"
      ironic_endpoint: "http://172.00.0.3:6385/v1/"
      ironic_inspector_endpoint: "http://172.00.0.3:5150/v1/"
      cache_url: "http://192.168.111.1/images"
      rhcos_image_url: "https://releases-art-
rhcos.svc.ci.openshift.org/art/storage/releases/rhcos-
4.3/43.81.201911192044.0/x86_64/rhcos-43.81.201911192044.0-
openstack.x86_64.qcow2.gz"
```

- ClusterDeployment with resource manifest **ConfigMap** referenced
  The resource manifest **ConfigMap** is referenced under **spec.provisioning.manifestsConfigMapRef**.

```
apiVersion: hive.openshift.io/v1
```

```
kind: ClusterDeployment
metadata:
  name: <my-baremetal-cluster>
  namespace: <mynamespace>
  annotations:
    hive.openshift.io/try-install-once: "true"
spec:
  baseDomain: test.example.com
  clusterName: <my-baremetal-cluster>
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    baremetal:
      libvirtSSHPrivateKeySecretRef:
        name: provisioning-host-ssh-private-key
  provisioning:
    installConfigSecretRef:
      name: <my-baremetal-cluster-install-config>
    sshPrivateKeySecretRef:
      name: <my-baremetal-hosts-ssh-private-key>
    manifestsConfigMapRef:
      name: <my-baremetal-cluster-install-manifests>
    imageSetRef:
      name: <my-clusterimageset>
    sshKnownHosts:
    - "10.1.8.90 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXvVVVKUYVkuyvkuygkuyTCYTytfkufTYAAAAIbmlzdHAyNTYAAABBBKWjJR
zeUVuZs4yxSy4eu45xiANFIIbwE3e1aPzGD58x/NX7Yf+S8eFKq4RrsfSaK2hVJyJjvVIhUsU9z2s
BJP8="
  pullSecretRef:
    name: <my-baremetal-cluster-pull-secret>
```

1.6.2. Creating a cluster on Amazon Web Services

You can use the Red Hat Advanced Cluster Management for Kubernetes console to create a Red Hat OpenShift Container Platform cluster on Amazon Web Services (AWS).

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on AWS in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.6.2.1. Prerequisites

See the following prerequisites before creating a cluster on AWS:

- You must have a deployed Red Hat Advanced Cluster Management for Kubernetes hub cluster.
- You need Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster so it can create the Kubernetes cluster on Amazon Web Services.

- You need an AWS credential. See Creating a credential for Amazon Web Services for more information.
- You need a configured domain in AWS. See Configuring an AWS account for instructions on how to configure a domain.
- You must have Amazon Web Services (AWS) login credentials, which include user name, password, access key ID, and secret access key. See Understanding and Getting Your Security Credentials.
- You must have an OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: If you change your cloud provider access key, you must manually update the provisioned cluster access key. For more information, see the known issue, Automatic secret updates for provisioned clusters is not supported.

1.6.2.2. Creating your cluster with the console

To create a cluster from the Red Hat Advanced Cluster Management console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Amazon Web Services for more information.

The name of the cluster is used in the hostname of the cluster.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.6.2.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured with your AWS account, that value is populated in the field. You can change the value by overwriting it. This name is used in the hostname of the cluster. See Configuring an AWS account for more information.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list

of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following fields:

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.
- Zones: Specify where you want to run your control plane pools. You can select multiple zones within the region for a more distributed group of control plane nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.
- Instance type: Specify the instance type for your control plane node. You can change the type and size of your instance after it is created.
- Root storage: Specify the amount of root storage to allocate for the cluster.

You can create zero or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The optional information includes the following fields:

- Zones: Specify where you want to run your worker pools. You can select multiple zones within the region for a more distributed group of nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.
- Instance type: Specify the instance type of your worker pools. You can change the type and size of your instance after it is created.
- Node count: Specify the node count of your worker pool. This setting is required when you define a worker pool.
- Root storage: Specify the amount of root storage allocated for your worker pool. This setting is required when you define a worker pool.

Networking details are required for your cluster, and multiple networks are required for using IPv6. You can add an additional network by clicking **Add network**.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: Specify the secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can select **YAML: On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.3. Creating a cluster on Microsoft Azure

You can use the Red Hat Advanced Cluster Management for Kubernetes console to deploy a Red Hat OpenShift Container Platform cluster on Microsoft Azure or on Microsoft Azure Government.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on Azure in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.6.3.1. Prerequisites

See the following prerequisites before creating a cluster on Azure:

- You must have a deployed Red Hat Advanced Cluster Management for Kubernetes hub cluster.
- You need Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster so it can create the Kubernetes cluster on Azure or Azure Government
- You need an Azure credential. See Creating a credential for Microsoft Azure for more information.
- You need a configured domain in Azure or Azure Government. See Configuring a custom domain name for an Azure cloud service for instructions on how to configure a domain.
- You need Azure login credentials, which include user name and password. See the Microsoft Azure Portal.
- You need Azure service principals, which include **clientId**, **clientSecret**, and **tenantId**. See azure.microsoft.com.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: If you change your cloud provider access key, you must manually update the provisioned cluster access key. For more information, see the known issue, Automatic secret updates for provisioned clusters is not supported.

1.6.3.2. Creating your cluster with the console

To create a cluster from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Microsoft Azure for more information.

The name of the cluster is used in the hostname of the cluster.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.6.3.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your Azure account, that value is populated in that field. You can change the value by overwriting it. See Configuring a custom domain name for an Azure cloud service for more information. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The Node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following optional fields:

- Region: Specify a region where you want to run your node pools. You can select multiple zones within the region for a more distributed group of control plane nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.
- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.
- Instance type and Root storage allocation (required) for your control plane pool. You can change the type and size of your instance after it is created.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes the following fields:

- Zones: Specifies here you want to run your worker pools. You can select multiple zones within the region for a more distributed group of nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.
- Instance type: You can change the type and size of your instance after it is created.

You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.4. Creating a cluster on Google Cloud Platform

Follow the procedure to create a Red Hat OpenShift Container Platform cluster on Google Cloud Platform (GCP). For more information about GCP, see Google Cloud Platform.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on GCP in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.6.4.1. Prerequisites

See the following prerequisites before creating a cluster on GCP:

- You must have a deployed Red Hat Advanced Cluster Management for Kubernetes hub cluster.

- You need Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster so it can create the Kubernetes cluster on GCP.
- You must have a GCP credential. See Creating a credential for Google Cloud Platform for more information.
- You must have a configured domain in GCP. See Setting up a custom domain for instructions on how to configure a domain.
- You need your GCP login credentials, which include user name and password.
- You must have an OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: If you change your cloud provider access key, you must manually update the provisioned cluster access key. For more information, see the known issue, Automatic secret updates for provisioned clusters is not supported.

1.6.4.2. Creating your cluster with the console

To create clusters from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Google Cloud Platform for more information.

The name of your cluster is used in the hostname of the cluster. There are some restrictions that apply to naming your GCP cluster. These restrictions include not beginning the name with **goog** or containing a group of letters and numbers that resemble **google** anywhere in the name. See Bucket naming guidelines for the complete list of restrictions.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.6.4.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential for your GCP account, that value is populated in the field. You can change the value by overwriting it. See Setting up a custom domain for more information. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The Node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following fields:

- Region: Specify a region where you want to run your control plane pools. A closer region might provide faster performance, but a more distant region might be more distributed.
- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.
- Instance type: You can change the type and size of your instance after it is created.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes the following fields:

- Instance type: You can change the type and size of your instance after it is created.
- Node count: This setting is required when you define a worker pool.

The networking details are required, and multiple networks are required for using IPv6 addresses. You can add an additional network by clicking **Add network**.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can select **YAML: On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.5. Creating a cluster on VMware vSphere

You can use the Red Hat Advanced Cluster Management for Kubernetes console to deploy a Red Hat OpenShift Container Platform cluster on VMware vSphere.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on vSphere in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.6.5.1. Prerequisites

See the following prerequisites before creating a cluster on vSphere:

- You must have a Red Hat Advanced Cluster Management hub cluster that is deployed on OpenShift Container Platform version 4.6 or later.
- You need Internet access for your Red Hat Advanced Cluster Management hub cluster so it can create the Kubernetes cluster on vSphere.
- You need a vSphere credential. See Creating a credential for VMware vSphere for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets.
- You must have the following information for the VMware instance where you are deploying:
  - Required static IP addresses for API and Ingress instances
  - DNS records for:
    - **api.<cluster_name>.<base_domain>** which must point to the static API VIP
    - ***.apps.<cluster_name>.<base_domain>** which must point to the static IP address for Ingress VIP

Note: When creating a cluster by using the VMware vSphere or Red Hat OpenStack Platform providers and disconnected installation, if a certificate is required to access the mirror registry, you must enter it in the *Additional trust bundle* field of your credential in the *Configuration for disconnected installation section*. You cannot enter them in the cluster creation console editor.

1.6.5.2. Creating your cluster with the console

To create a cluster from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for VMware vSphere for more information about creating a credential.

The name of your cluster is used in the hostname of the cluster.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.6.5.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base domain associated with the selected credential that you configured for your vSphere account, that value is populated in the field. You can change the value by overwriting it. See Installing a cluster on vSphere with customizations for more information. This value must match the name that you used to create the DNS records listed in the prerequisites section. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images

Note: Only release images for OpenShift Container Platform versions 4.5.x and higher are supported.

The node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the *Architecture* field. View the following field description:

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes *Cores per socket*, *CPUs*, *Memory_min MB, _Disk size* in GiB, and *Node count*.

Networking information is required. Multiple networks are required for using IPv6. Some of the required networking information is included the following fields:

- vSphere network name: Specify the VMware vSphere network name.
- API VIP: Specify the IP address to use for internal API communication.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **api.** resolves correctly.
- Ingress VIP: Specify the IP address to use for ingress traffic.

Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **test.apps.** resolves correctly.

You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: Specify the secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: Provide a comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

You can define the disconnected installation image by clicking **Disconnected installation**. See Disconnected installation settings for cluster creation cannot be entered or are ignored if entered for more details on limitations.

You can click **Add automation template** to create a template.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.6. Creating a cluster on Red Hat OpenStack Platform

You can use the Red Hat Advanced Cluster Management for Kubernetes console to deploy a Red Hat OpenShift Container Platform cluster on Red Hat OpenStack Platform.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on OpenStack in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.6.6.1. Prerequisites

See the following prerequisites before creating a cluster on Red Hat OpenStack Platform:

- You must have a Red Hat Advanced Cluster Management hub cluster that is deployed on OpenShift Container Platform version 4.6, or later.
- You need Internet access for your Red Hat Advanced Cluster Management hub cluster so it can create the Kubernetes cluster on Red Hat OpenStack Platform.
- You must have a Red Hat OpenStack Platform credential. See Creating a credential for Red Hat OpenStack Platform for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets.
- You need the following information for the Red Hat OpenStack Platform instance where you are deploying:
  - Flavor name for the control plane and worker instances; for example, **m1.xlarge**
  - Network name for the external network to provide the floating IP addresses
  - Required floating IP addresses for API and ingress instances
  - DNS records for:
    - **api.<cluster_name>.<base_domain>**, which must point to the floating IP address for the API
    - ***.apps.<cluster_name>.<base_domain>**, which must point to the floating IP address for ingress

1.6.6.2. Creating your cluster with the console

To create a cluster from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Red Hat OpenStack Platform for more information.

The name of the cluster is used in the hostname of the cluster. The name must contain fewer than 15 characters. This value must match the name that you used to create the DNS records listed in the credential prerequisites section.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.6.6.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct

permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your Red Hat OpenStack Platform account, that value is populated in the field. You can change the value by overwriting it. See Managing domains in the Red Hat OpenStack Platform documentation for more information. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images. Only release images for OpenShift Container Platform versions 4.6.x and higher are supported.

The node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following fields:

- **Optional** Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.
- Instance type for your control plane pool: You can change the type and size of your instance after it is created.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes the following fields:

- Instance type: You can change the type and size of your instance after it is created.
- Node count: Specify the node count for your worker pool. This setting is required when you define a worker pool.

Networking details are required for your cluster. You must provide the values for one or more networks for an IPv4 network. For an IPv6 network, you must define more than one network.

You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: Define a comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.

- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

You can define the disconnected installation image by clicking **Disconnected installation**. See Disconnected installation settings for cluster creation cannot be entered or are ignored if entered for more details on limitations.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.7. Creating a cluster on Red Hat Virtualization

You can use the Red Hat Advanced Cluster Management for Kubernetes console to create a Red Hat OpenShift Container Platform cluster on Red Hat Virtualization.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on RHV in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.6.7.1. Prerequisites

See the following prerequisites before creating a cluster on Red Hat Virtualization:

- You must have a deployed Red Hat Advanced Cluster Management for Kubernetes hub cluster.
- You need Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster so it can create the Kubernetes cluster on Red Hat Virtualization.
- You need a Red Hat Virtualization credential. See Creating a credential for Red Hat Virtualization for more information.
- You need a configured domain and virtual machine proxy for the oVirt Engine virtual machines. See Installing on RHV in the Red Hat OpenShift Container Platform documentation for instructions on how to configure a domain.
- You must have Red Hat Virtualization login credentials, which include your Red Hat Customer Portal username and password.
- You need an OpenShift Container Platform image pull secret. You can download your pull secret from: Pull secret. See Using image pull secrets for more information about pull secrets.

Note: If you change your cloud provider access key, you must manually update the provisioned cluster access key. For more information, see the known issue, Automatic secret updates for provisioned clusters is not supported.

1.6.7.2. Creating your cluster with the console

To create a cluster from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Red Hat Virtualization for more information.

The name of your cluster is used in the hostname of the cluster.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.6.7.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your Red Hat Virtualization account, that value is populated in that field. You can overwrite the value to change it.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The information for your node pools includes the number of Cores, Sockets, Memory, and Disk size for the the control plane pool. The three control plane nodes share the management of the cluster activity. The information includes the *Architecture* field. View the following field description:

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.

The worker pool information requires the pool name, number of cores, memory allocation, disk size allocation, and node count for your worker pools. Your worker nodes within the worker pool can be in a single worker pool, or distributed across multiple worker pools.

The following networking details are required from your preconfigured oVirt environment.

- oVirt network name
- API VIP: Specify the IP address to use for internal API communication.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **api.** resolves correctly.
- Ingress VIP: Specify the IP address to use for ingress traffic.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **test.apps.** resolves correctly.
- Network type: The default value is **OpenShiftSDN**. OVNKubernetes is the required setting for using IPv6.
- Cluster network CIDR: This is a number and list of IP addresses that can be used for the pod IP addresses. This block must not overlap another network block. The default value is **10.128.0.0/14**.
- Network host prefix: Set the subnet prefix length for each node. The default value is **23**.
- Service network CIDR: Provide a block of IP addresses for services. This block must not overlap another network block. The default value is **172.30.0.0/16**.
- Machine CIDR: Provide a block of IP addresses that are used by the OpenShift Container Platform hosts. This block must not overlap another network block. The default value is **10.0.0.0/16**.
  You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: Specify the secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: Provide a comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.8. Creating a cluster on bare metal

You can use the Red Hat Advanced Cluster Management for Kubernetes console to create a Red Hat OpenShift Container Platform cluster in a bare metal environment.

Deprecation notice: The procedure for creating bare metal clusters using bare metal assets is deprecated. Bare metal assets will be removed from a future release.

When you create a cluster, note that the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on bare metal in the OpenShift Container Platform documentation for more information.

- Prerequisites
- Creating a bare metal cluster

1.6.8.1. Prerequisites

See the following prerequisites before creating a cluster in a bare metal environment:

- You must have a deployed Red Hat Advanced Cluster Management for Kubernetes hub cluster on OpenShift Container Platform version 4.6 or later.
- You need Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster (connected), or a connection to an internal or mirror registry that has a connection to the Internet (disconnected) to retrieve the required images for creating the cluster.
- You need a temporary external KVM host that runs a bootstrap virtual machine, which is used to create a Hive cluster. See Preparing a provisioning host for more information.
- The deployed Red Hat Advanced Cluster Management for Kubernetes hub cluster must be able to route to the provisioning network.
- You need your bare metal server login credentials, which includes the libvirt URI from the bootstrap virtual machine in the previous item, the SSH Private Key, and a list of SSH known hosts. See Setting up the environment for an OpenShift installation for more information.
- You need a configured bare metal credential. See Creating a credential for bare metal for more information.
- You must have login credentials for your bare metal environment, which include user name, password, and Baseboard Management Controller Address.
- You need a configured bare metal asset, if you are are enabling certificate verification. See Creating and modifying bare metal assets for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets for more information.
  Notes:
  - The bare metal asset, managed bare metal cluster, and its related secret must be in the same namespace.
  - If you change your cloud provider access key, you must manually update the provisioned cluster access key. For more information, see the known issue, Automatic secret updates for provisioned clusters is not supported.

- When you create a cluster by using the Bare metal provider and a disconnected installation, you must store all your settings in the credential in the *Configuration for disconnected installation* section. You cannot enter them in the cluster creation console editor.

1.6.8.2. Creating a bare metal cluster

To create a cluster from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for bare metal for more information about creating a credential.

For a bare metal cluster, the name of the cluster cannot be an arbitrary name. It is associated with the cluster URL. Make sure that the cluster name that you use is consistent with your DNS and network setup.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

The base domain of your provider is used to create routes to your Red Hat OpenShift Container Platform cluster components. It is configured in the DNS of your cluster provider as a Start of Authority (SOA) record. This name is used in the hostname of the cluster.

If there is already a base DNS domain that is associated with the selected credential that you configured for your bare metal provider account, that value is populated in that field. You can change the value by overwriting it, but you cannot change the name after the cluster is created. See Installing on bare metal in the OpenShift Container Platform documentation for more information.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The list of hosts is compiled from the existing bare metal assets, and are associated with your credential. Ensure that you are running the latest firmware on the bare metal hosts, or the provision might fail. You must select a minimum of three bare metal assets that are on the same bridge networks as the hypervisor. If you do not have any bare metal assets created, then you can create or import them before you continue with the creation process by selecting **Import assets**. For more information about creating bare metal assets, see Creating and modifying bare metal assets . Alternatively, you can select **Disable certificate verification** to bypass the requirement.

The following table shows the networking options and their descriptions:

Parameter	Description	Required or Optional
Provisioning network CIDR	The CIDR for the network to use for provisioning. The example format is: 172.30.0.0/16.	Required
Provisioning network interface	The name of the network interface on the control plane nodes that are connected to the provisioning network.	Required
Provisioning network bridge	The name of the bridge on the hypervisor that is attached to the provisioning network.	Required
External network bridge	The name of the bridge of the hypervisor that is attached to the external network.	Required
API VIP	The Virtual IP to use for internal API communication. The DNS must be pre-configured with an A/AAAA or CNAME record so the **api.<cluster_name>.<Base DNS domain>** path resolves correctly.	Required
Ingress VIP	The Virtual IP to use for ingress traffic. The DNS must be pre-configured with an A/AAAA or CNAME record so the ***.apps.<cluster_name>.<Base DNS domain>** path resolves correctly.	Optional
Network type	The pod network provider plug-in to deploy. Only the OpenShiftSDN plug-in is supported on OpenShift Container Platform 4.3. The OVNKubernetes plug-in is available as a Technology Preview on OpenShift Container Platform versions 4.3, 4.4, and 4.5. It is generally available on OpenShift Container Platform version 4.6, and later. OVNKubernetes must be used with IPv6. The default value is **OpenShiftSDN**.	Required

Parameter	Description	Required or Optional
Cluster network CIDR	A block of IP addresses from which pod IP addresses are allocated. The OpenShiftSDN network plug-in supports multiple cluster networks. The address blocks for multiple cluster networks must not overlap. Select address pools large enough to fit your anticipated workload. The default value is 10.128.0.0/14.	Required
Network host prefix	The subnet prefix length to assign to each individual node. For example, if hostPrefix is set to 23, then each node is assigned a /23 subnet out of the given CIDR, allowing for 510 (2^(32-23)-2) pod IP addresses. The default is 23.	Required
Service network CIDR	A block of IP addresses for services. OpenShiftSDN allows only one serviceNetwork block. The address must not overlap any other network block. The default value is 172.30.0.0/16.	Required
Machine CIDR	A block of IP addresses used by the OpenShift Container Platform hosts. The address block must not overlap any other network block. The default value is 10.0.0.0/16.	Required

You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.

- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can select **YAML: On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured with the management of Red Hat Advanced Cluster Management.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.9. Creating a cluster in an on-premises environment

You can use the Red Hat Advanced Cluster Management for Kubernetes console to create an on-premises Red Hat OpenShift Container Platform cluster. This process is recommended instead of the bare metal process that is deprecated.

Best practice: Use this procedure for creating a single-node OpenShift (SNO) cluster. You can create a single-node OpenShift cluster on VMware vSphere, Red Hat OpenStack, Red Hat Virtualization Platform, and in a bare metal environment. There is no platform integration with the platform where you install the cluster, as the platform value is set to **platform=none**. A single-node OpenShift cluster contains only a single node, which hosts the control plane services and the user workloads. This configuration can be helpful when you want to minimize the resource footprint of the cluster.

You can also test the procedure of provisioning multiple single-node OpenShift clusters on edge resources by using the zero touch provisioning feature, which is a Technology Preview feature that is available with Red Hat OpenShift Container Platform. For more information about that procedure, see Deploying distributed units at scale in a disconnected environment in the OpenShift Container Platform documentation.

- Prerequisites
- Creating your cluster with the console

1.6.9.1. Prerequisites

See the following prerequisites before creating a cluster in an on-premises environment:

- You must have a deployed Red Hat Advanced Cluster Management hub cluster on OpenShift Container Platform version 4.9, or later.
- You need a configured infrastructure environment with configured hosts. See Creating an infrastructure environment for more information.
- You must have Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster (connected), or a connection to an internal or mirror registry that has a connection to the Internet (disconnected) to retrieve the required images for creating the cluster.
- You need a configured on-premises credential. See Creating a credential for an on-premises environment for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets for more information.

1.6.9.2. Creating your cluster with the console

To create a cluster from the Red Hat Advanced Cluster Management for Kubernetes console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

The following options are available for your assisted installation:

- **Use existing discovered hosts**: Select your hosts from a list of hosts that are in an existing infrastructure environment.
- **Discover new hosts**: Discover hosts that are not already in an existing infrastructure environment. Discover your own hosts, rather than using one that is already in an infrastructure environment.

If you need to create a credential, see Creating a credential for an on-premises environment for more information.

The name for your cluster is used in the hostname of the cluster.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your provider account, that value is populated in that field. You can change the value by overwriting it, but this setting cannot be changed after the cluster is created. The base domain of your provider is used to create routes to your Red Hat OpenShift Container Platform cluster components. It is configured in the DNS of your cluster provider as a Start of Authority (SOA) record.

The **OpenShift version** identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

When you select an OpenShift version that is 4.9 or later, an option to select **Install single node OpenShift (SNO)** is displayed. A single-node OpenShift cluster contains a single node which hosts the control plane services and the user workloads. You cannot add additional nodes to a single-node OpenShift cluster after it is created.

If you want your cluster to be a single-node OpenShift cluster, select the single-node OpenShift option.

Note: The single-node OpenShift control plane requires 8 CPU cores, while a control plane node for a multinode control plane cluster only requires 4 CPU cores.

After you review and save the cluster, your cluster is saved as a draft cluster. You can close the creation process and finish the process later by selecting the cluster name on the *Clusters* page.

If you are using existing hosts, select whether you want to select the hosts yourself, or if you want them to be selected automatically. The number of hosts is based on the number of nodes that you selected. For example, a SNO cluster only requires one host, while a standard three-node cluster requires three hosts.

The locations of the available hosts that meet the requirements for this cluster are displayed in the list of *Host locations*. For distribution of the hosts and a more high-availability configuration, select multiple locations.

If you are discovering new hosts with no existing infrastructure environment, complete the steps in Adding hosts to an infrastructure environment beginning with step 4 to define your hosts.

After the hosts are bound, and the validations pass, complete the networking information for your cluster by adding the following IP addresses:

- API VIP: Specifies the IP address to use for internal API communication.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **api.** resolves correctly.

- Ingress VIP: Specifies the IP address to use for ingress traffic.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **test.apps.** resolves correctly.

You can view the status of the installation on the *Clusters* navigation page.

Continue with Accessing your cluster for instructions for accessing your cluster.

1.6.10. Hibernating a created cluster (Technology Preview)

You can hibernate a cluster that was created using Red Hat Advanced Cluster Management for Kubernetes to conserve resources. A hibernating cluster requires significantly fewer resources than one that is running, so you can potentially lower your provider costs by moving clusters in and out of a hibernating state. This feature only applies to clusters that were created by Red Hat Advanced Cluster Management in the following environments:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform

1.6.10.1. Hibernate a cluster by using the console

To use the Red Hat Advanced Cluster Management console to hibernate a cluster that was created by Red Hat Advanced Cluster Management, complete the following steps:

1. From the Red Hat Advanced Cluster Management navigation menu, select **Infrastructure** > **Clusters**. Ensure that the *Manage clusters* tab is selected.

2. Select **Hibernate cluster** from the the *Options* menu for the cluster. **Note:** If the *Hibernate cluster* option is not available, you cannot hibernate the cluster. This can happen when the cluster is imported, and not created by Red Hat Advanced Cluster Management.

The status for the cluster on the *Clusters* page is **Hibernating** when the process completes.

Tip: You can hibernate multiple clusters by selecting the clusters that you want to hibernate on the *Clusters* page, and selecting **Actions** > **Hibernate clusters**.

Your selected cluster is hibernating.

1.6.10.2. Hibernate a cluster by using the CLI

To use the CLI to hibernate a cluster that was created by Red Hat Advanced Cluster Management, complete the following steps:

1. Enter the following command to edit the settings for the cluster that you want to hibernate:

   ```
   oc edit clusterdeployment <name-of-cluster> -n <namespace-of-cluster>
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

2. Change the value for **spec.powerState** to **Hibernating**.

3. Enter the following command to view the status of the cluster:

   ```
   oc get clusterdeployment <name-of-cluster> -n <namespace-of-cluster> -o yaml
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

   When the process of hibernating the cluster is complete, the value of the type for the cluster is **type=Hibernating**.

Your selected cluster is hibernating.

1.6.10.3. Resuming normal operation of a hibernating cluster by using the console

To resume normal operation of a hibernating cluster by using the Red Hat Advanced Cluster Management console, complete the following steps:

1. From the Red Hat Advanced Cluster Management navigation menu, select **Infrastructure** > **Clusters**. Ensure that the *Manage clusters* tab is selected.

2. Select **Resume cluster** from the the *Options* menu for the cluster that you want to resume.

The status for the cluster on the *Clusters* page is **Ready** when the process completes.

Tip: You can resume multiple clusters by selecting the clusters that you want to resume on the *Clusters* page, and selecting **Actions** > **Resume clusters**.

Your selected cluster is resuming normal operation.

1.6.10.4. Resuming normal operation of a hibernating cluster by using the CLI

To resume normal operation of a hibernating cluster by using the CLI, complete the following steps:

1. Enter the following command to edit the settings for the cluster:

   ```
   oc edit clusterdeployment <name-of-cluster> -n <namespace-of-cluster>
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

2. Change the value for **spec.powerState** to **Running**.

3. Enter the following command to view the status of the cluster:

   ```
   oc get clusterdeployment <name-of-cluster> -n <namespace-of-cluster> -o yaml
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

   When the process of resuming the cluster is complete, the value of the type for the cluster is **type=Running**.

Your selected cluster is resuming normal operation.

1.7. IMPORTING A TARGET MANAGED CLUSTER TO THE HUB CLUSTER

You can import clusters from different Kubernetes cloud providers. After you import, the targeted cluster becomes a managed cluster for the Red Hat Advanced Cluster Management for Kubernetes hub cluster. Unless otherwise specified, complete the import tasks anywhere where you can access the hub cluster and the targeted managed cluster.

A hub cluster cannot manage *any* other hub cluster, but can manage itself. The hub cluster is configured to automatically be imported and self-managed. You do not need to manually import the hub cluster.

However, if you remove a hub cluster and try to import it again, you need to add the **local-cluster:true** label.

Choose from the following instructions to set up your managed cluster, either from the console or from the CLI:

Required user type or access level Cluster administrator

- Importing an existing cluster with the console
- Importing a managed cluster with the CLI
- Modifying the klusterlet add-ons settings of your cluster

1.7.1. Importing an existing cluster with the console

After you install Red Hat Advanced Cluster Management for Kubernetes, you are ready to import a cluster to manage. You can import from both the console and the CLI.

Follow this procedure to import from the console. You need your terminal for authentication during this procedure.

- Prerequisites
- Importing a cluster
- Removing a cluster

1.7.1.1. Prerequisites

- You need a Red Hat Advanced Cluster Management for Kubernetes hub cluster that is deployed. If you are importing bare metal clusters, you must have the hub cluster installed on Red Hat OpenShift Container Platform version 4.8 or later.
- You need a cluster that you want to manage and Internet connectivity.
- Install **kubectl**. To install **kubectl**, see *Install and Set Up kubectl* in the Kubernetes documentation.
- You need the **base64** command line tool.
- **Note:** If you are importing a cluster that was not created by OpenShift Container Platform, you need a **multiclusterhub.spec.imagePullSecret** defined. This secret might be created when Red Hat Advanced Cluster Management was installed.
  If you need to create a new one, complete the following steps:

  1. Download your Kubernetes pull secret from cloud.redhat.com.
  2. Add the pull secret to the namespace of your hub cluster.
  3. Run the following command to create a new secret in the namespace of your hub cluster:

     ```
     oc create secret generic pull-secret -n <open-cluster-management> --from-file=.dockerconfigjson=<path-to-pull-secret> --type=kubernetes.io/dockerconfigjson
     ```

     Replace **open-cluster-management** with the name of the namespace of your hub cluster. The default namespace of the hub cluster is **open-cluster-management**.

     Replace **path-to-pull-secret** with the path to the pull secret that you downloaded.

     The secret is automatically copied to the managed cluster when it is imported.

     See Using image pull secrets or Understanding and creating service accounts for more information about pull secrets.

     See Custom image pull secret for more information about how to define this secret.

- Ensure the agent is deleted on the cluster that you want to import. The **open-cluster-management-agent** and **open-cluster-management-agent-addon** namespaces must be removed to avoid errors.
- For importing in a Red Hat OpenShift Dedicated environment, see the following notes:

- You must have the hub cluster deployed in a Red Hat OpenShift Dedicated environment.
- The default permission in Red Hat OpenShift Dedicated is dedicated-admin, but that does not contain all of the permissions to create a namespace. You must have **cluster-admin** permissions to import and manage a cluster with Red Hat Advanced Cluster Management for Kubernetes.

Required user type or access level Cluster administrator

1.7.1.2. Importing a cluster

You can import existing clusters from the Red Hat Advanced Cluster Management for Kubernetes console for each of the available cloud providers.

Note: A hub cluster cannot manage a different hub cluster. A hub cluster is set up to automatically import and manage itself, so you do not have to manually import a hub cluster to manage itself.

1. From the navigation menu, select **Infrastructure** > **Clusters**.
2. In the *Managed clusters* tab, click **Import cluster**.
3. Provide a name for the cluster. By default, the namespace is used for the cluster name and namespace.

Important: When you create a cluster, the Red Hat Advanced Cluster Management controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

1. Specify a *Cluster set*, if you want to add it to an existing cluster set on which you have **cluster-admin** privileges. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have cluster set options to select.
   Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

2. **Optional:** Add any *Additional labels*.
   Note: If you import a Red Hat OpenShift Dedicated cluster and do not specify a vendor by adding a label for **vendor=OpenShiftDedicated**, or if you add a label for **vendor=auto-detect**, a **managed-by=platform** label is automatically added to the cluster. You can use this added label to identify the cluster as a Red Hat OpenShift Dedicated cluster and retrieve the Red Hat OpenShift Dedicated clusters as a group.

3. Select the *import mode* that you want to use to identify the cluster that you are importing from the following options:

   - **Run import commands manually**: Generate import commands that you can copy and run, based on the information that you provided. Click **Save import and generate code** to generate the command that you use to deploy the **open-cluster-management-agent-addon**. A confirmation message is displayed.

     a. In the *Import an existing cluster* window, select **Copy command** to copy the generated command and token to the clipboard.
     Important: The command contains pull secret information that is copied to each of the imported clusters. Anyone who can access the imported clusters can also view the pull

secret information. Consider creating a secondary pull secret at https://cloud.redhat.com/ or create a service account to protect your personal credentials.

b. Log in to the managed cluster that you want to import.

c. **For the Red Hat OpenShift Dedicated environment only:**Complete the following steps:

   i. Create the **open-cluster-management-agent** and **open-cluster-management** namespaces or projects on the managed cluster.

   ii. Find the klusterlet operator in the OpenShift Container Platform catalog.

   iii. Install it in the **open-cluster-management** namespace or project that you created. **Important:** Do not install the operator in the **open-cluster-management-agent** namespace.

   iv. Extract the bootstrap secret from the import command by completing the following steps:

      A. Generate the import command:

         I. Select **Infrastructure** > **Clusters** from the Red Hat Advanced Cluster Management console main navigation.

         II. Select **Add a cluster** > **Import an existing cluster**.

         III. Add the cluster information, and select **Save import and generate code**

      B. Copy the import command.

      C. Paste the import command into a file that you create named **import-command**.

      D. Run the following command to insert the content into the new file:

```
cat import-command | awk '{split($0,a,"&&"); print a[3]}' | awk '{split($0,a,"|"); print a[1]}' | sed -e "s/^ echo //" | base64 -d
```

      E. Find and copy the secret with the name **bootstrap-hub-kubeconfig** in the output.

      F. Apply the secret to the **open-cluster-management-agent** namespace on the managed cluster.

      G. Create the klusterlet resource using the example in the installed operator, the clusterName should be changed the same name as cluster name that was set during the import.
      Note: When the **managedcluster** resource is successfully registered to the hub, there are two klusterlet operators installed. One klusterlet operator is in the **open-cluster-management** namespace, and the other is in the **open-cluster-management-agent** namespace. Multiple operators does not affect the function of the klusterlet.

d. **For cluster imports that are not in the Red Hat OpenShift Dedicated environment** Complete the following steps:

i. If necessary, configure your **kubectl** commands for your managed cluster. See Supported providers to learn how to configure your **kubectl** command line interface.

ii. To deploy the **open-cluster-management-agent-addon** to the managed cluster, run the command and token that you copied.

e. Select **View cluster** to view a summary of your cluster in the *Overview* page.

- **Enter your server URL and API token for the existing cluster**: Provide the server URL and API token of the cluster that you are importing.
- **Kubeconfig**: Copy and paste the content of the **kubeconfig** file of the cluster that you are importing.

4. **Optional:** Configure the **Cluster API address** that is on the cluster details page by configuring the URL that is displayed in the table when you run the **oc get managedcluster** command.

a. Log in to your hub cluster with an ID that has **cluster-admin** permissions.

b. Configure your **kubectl** for your targeted managed cluster. See Supported providers to learn how to configure your **kubectl**.

c. Edit the managed cluster entry for the cluster that you are importing by entering the following command:

```
oc edit managedcluster <cluster-name>
```

Replace ***cluster-name*** with the name of the managed cluster.

d. Add the **ManagedClusterClientConfigs** section to the **ManagedCluster** spec in the YAML file, as shown in the following example:

```
spec:
  hubAcceptsClient: true
  managedClusterClientConfigs:
  - url: https://multicloud-console.apps.new-managed.dev.redhat.com
```

Replace the value of the URL with the URL that provides external access to the managed cluster that you are importing.

Your cluster is imported. You can import another by selecting **Import another**.

1.7.1.3. Removing an imported cluster

Complete the following procedure to remove an imported cluster and the **open-cluster-management-agent-addon** that was created on the managed cluster.

On the *Clusters* page, click **Actions** > **Detach cluster** to remove your cluster from management.

Note: If you attempt to detach the hub cluster, which is named **local-cluster**, be aware that the default setting of **disableHubSelfManagement** is **false**. This setting causes the hub cluster to reimport itself and manage itself when it is detached and it reconciles the **MultiClusterHub** controller. It might take hours for the hub cluster to complete the detachment process and reimport. If you want to reimport the hub cluster without waiting for the processes to finish, you can enter the following command to restart the **multiclusterhub-operator** pod and reimport faster:

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
```

You can change the value of the hub cluster to not import automatically by changing the **disableHubSelfManagement** value to **true**. For more information, see the disableHubSelfManagement topic.

1.7.2. Importing a managed cluster with the CLI

After you install Red Hat Advanced Cluster Management for Kubernetes, you are ready to import a cluster to manage by using the Red Hat OpenShift Container Platform CLI. You can import a cluster using the **kubeconfig** file of the cluster that you are importing, or you can run import commands manually on the cluster you are importing. Both procedures are documented.

- Prerequisites
- Supported architecture
- Prepare for import
- Importing the cluster with the auto import secret
- Importing the cluster with the manual commands
- Importing the klusterlet add-on

Important: A hub cluster cannot manage a different hub cluster. A hub cluster is set up to automatically import and manage itself. You do not have to manually import a hub cluster to manage itself.

However, if you remove a hub cluster and try to import it again, you need to add the **local-cluster:true** label.

1.7.2.1. Prerequisites

- You need a Red Hat Advanced Cluster Management for Kubernetes hub cluster that is deployed. If you are importing bare metal clusters, you must have the hub cluster installed on Red Hat OpenShift Container Platform version 4.6 or later.
- You need a separate cluster that you want to manage that has Internet connectivity.
- You need the Red Hat OpenShift Container Platform CLI version 4.6 or later, to run **oc** commands. See Getting started with the OpenShift CLI for information about installing and configuring the Red Hat OpenShift Container Platform CLI, **oc**.
- You need to install the Kubernetes CLI, **kubectl**. To install **kubectl**, see *Install and Set Up kubectl* in the Kubernetes documentation.
  Note: Download the installation file for CLI tools from the console.
- If you are importing a cluster that was not created by OpenShift Container Platform, you need a **multiclusterhub.spec.imagePullSecret** defined. This secret might have been created when Red Hat Advanced Cluster Management for Kubernetes was installed. See Custom Image Pull Secret for more information about defining the secret.

1.7.2.2. Supported architectures

- Linux (x86_64, s390x, ppc64le)

- macOS

1.7.2.3. Prepare for import

1. Log in to your *hub cluster* by running the following command:

```
oc login
```

2. Run the following command on the hub cluster to create the project and namespace: **Note:** The cluster name that is defined in **CLUSTER_NAME** is also used as the cluster namespace in the YAML file and commands:

```
oc new-project ${CLUSTER_NAME}
```

Important: The **cluster.open-cluster-management.io/managedCluster** label is automatically added to and removed from a managed cluster namespace. Do not manually add it to or remove it from a managed cluster.

3. Create a file named **managed-cluster.yaml** with the following example content:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: ${CLUSTER_NAME}
  labels:
    cloud: auto-detect
    vendor: auto-detect
spec:
  hubAcceptsClient: true
```

When the values for **cloud** and **vendor** are set to **auto-detect**, Red Hat Advanced Cluster Management detects the cloud and vendor types automatically from the cluster that you are importing. You can optionally replace the values for **auto-detect** with with the cloud and vendor values for your cluster. See the following example:

```
cloud: Amazon
vendor: OpenShift
```

4. Apply the YAML file to the **ManagedCluster** resource by entering the following command:

```
oc apply -f managed-cluster.yaml
```

Continue with Importing the cluster with the auto import secret or Importing the cluster with the manual commands.

1.7.2.4. Importing the cluster with the auto import secret

To import with the auto import secret, you must create a secret that contains either a reference to the **kubeconfig** file of the cluster or the kube API server and token pair of the cluster.

1. Retrieve the **kubeconfig** file or the kube API server and token of the cluster that you are importing. See the documentation for your Kubernetes cluster to learn where to locate your **kubeconfig** file or your kube API server and token.

2. Create the **auto-import-secret.yaml** file in the ${CLUSTER_NAME} namespace.

   a. Create a YAML file named **auto-import-secret.yaml** that contains content that is similar to the following template:

```
apiVersion: v1
kind: Secret
metadata:
  name: auto-import-secret
  namespace: <cluster_name>
stringData:
  autoImportRetry: "5"
  # If you are using the kubeconfig file, add the following value for the kubeconfig file
  # that has the current context set to the cluster to import:
  kubeconfig: |- <kubeconfig_file>
  # If you are using the token/server pair, add the following two values instead of
  # the kubeconfig file:
  token: <Token to access the cluster>
  server: <cluster_api_url>
type: Opaque
```

   b. Apply the YAML file in the ${CLUSTER_NAME} namespace with the following command:

```
oc apply -f auto-import-secret.yaml
```

   Note: By default, the auto-import secret is used one time and deleted when the import process completes. If you want to keep the auto import secret, add **managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret** to the secret. You can add it by running the following command:

```
oc -n <cluster_name> annotate secrets auto-import-secret managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret=""
```

3. Validate the **JOINED** and **AVAILABLE** status for your imported cluster. Run the following command from the hub cluster:

```
oc get managedcluster ${CLUSTER_NAME}
```

4. Log in to the managed cluster by running the following command on the managed cluster:

```
oc login
```

5. Run the following command to validate the pod status on the cluster that you are importing:

```
oc get pod -n open-cluster-management-agent
```

Continue with Importing the klusterlet add-on.

1.7.2.5. Importing the cluster with the manual commands

Important: The import command contains pull secret information that is copied to each of the imported clusters. Anyone who can access the imported clusters can also view the pull secret information.

1. Obtain the **klusterlet-crd.yaml** file that was generated by the import controller on your hub cluster by running the following command:

```
oc get secret ${CLUSTER_NAME}-import -n ${CLUSTER_NAME} -o jsonpath=
{.data.crds\\.yaml} | base64 --decode > klusterlet-crd.yaml
```

2. Obtain the **import.yaml** file that was generated by the import controller on your hub cluster by running the following command:

```
oc get secret ${CLUSTER_NAME}-import -n ${CLUSTER_NAME} -o jsonpath=
{.data.import\\.yaml} | base64 --decode > import.yaml
```

Proceed with the following steps in the cluster that you are importing:

3. Log in to the managed cluster that you are importing by entering the following command:

```
oc login
```

4. Apply the **klusterlet-crd.yaml** that you generated in step 1 by running the following command:

```
oc apply -f klusterlet-crd.yaml
```

5. Apply the **import.yaml** file that you previously generated by running the following command:

```
oc apply -f import.yaml
```

6. Validate **JOINED** and **AVAILABLE** status for the cluster that you are importing. From the hub cluster, run the following command:

```
oc get managedcluster ${CLUSTER_NAME}
```

Continue with Importing the klusterlet add-on.

1.7.2.6. Importing the klusterlet add-on

You can create and apply the klusterlet add-on configuration file by completing the following procedure:

1. Create a YAML file that is similar to the following example:

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster_name>
  namespace: <cluster_name>
spec:
  applicationManager:
    enabled: true
  certPolicyController:
    enabled: true
  iamPolicyController:
    enabled: true
  policyController:
```

```
    enabled: true
  searchCollector:
    enabled: true
```

2. Save the file as **klusterlet-addon-config.yaml**.

3. Apply the YAML by running the following command:

```
oc apply -f klusterlet-addon-config.yaml
```

   The ManagedCluster-Import-Controller will generate a secret named **${CLUSTER_NAME}-import**. The **${CLUSTER_NAME}-import** secret contains the **import.yaml** that the user applies to a managed cluster to install klusterlet.

   Add-ons are installed after the cluster you are importing is **AVAILABLE**.

4. Validate the pod status of add-ons on the cluster you are importing by running the following command:

```
oc get pod -n open-cluster-management-agent-addon
```

Your cluster is now imported.

1.7.2.7. Removing an imported cluster with the CLI

To remove a cluster, run the following command:

```
oc delete managedcluster ${CLUSTER_NAME}
```

Replace **cluster_name** with the name of the cluster.

Your cluster is now removed.

1.7.3. Importing a cluster with a custom ManagedClusterImageRegistry CRD

There might be times when you need to override the image registry on the managed clusters that you are importing. You can do this by creating a **ManagedClusterImageRegistry** custom resource definition (CRD).

The **ManagedClusterImageRegistry** CRD is a namespace-scoped resource.

The **ManagedClusterImageRegistry** CRD specifies a set of managed clusters for a Placement to select, but needs different images from the custom image registry. After the managed clusters are updated with the new images, the following label is added to each managed cluster for identification: **open-cluster-management.io/image-registry=<namespace>.<managedClusterImageRegistryName>**.

The following example shows a **ManagedClusterImageRegistry** CRD:

```
apiVersion: imageregistry.open-cluster-management.io/v1alpha1
kind: ManagedClusterImageRegistry
metadata:
  name: <imageRegistryName>
  namespace: <namespace>
```

```
spec:
  placementRef:
    group: cluster.open-cluster-management.io
    resource: placements
    name: <placementName>
  pullSecret:
    name: <pullSecretName>
  registries:
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
```

In the **spec** section:

- Replace **placementName** with the name of a Placement in the same namespace that selects a set of managed clusters.
- Replace **pullSecretName** with the name of the pull secret that is used to pull images from the custom image registry.
- List the values for each of the **source** and **mirror** registries. Replace the **mirrored-image-registry-address** and **image-registry-address** with the value for each of the **mirror** and **source** values of the registries.
  - Example 1: To replace the source image registry named **registry.redhat.io/rhacm2** with **localhost:5000/rhacm2**, and **registry.redhat.io/multicluster-engine** with **localhost:5000/multicluster-engine**, use the following example:

    ```
    registries:
    - mirror: localhost:5000/rhacm2/
      source: registry.redhat.io/rhacm2
    - mirror: localhost:5000/multicluster-engine
      source: registry.redhat.io/multicluster-engine
    ```

  - Example 2: To replace the source image, **registry.redhat.io/rhacm2/registration-rhel8-operator** with **localhost:5000/rhacm2-registration-rhel8-operator**, use the following example:

    ```
    registries:
    - mirror: localhost:5000/rhacm2-registration-rhel8-operator
      source: registry.redhat.io/rhacm2/registration-rhel8-operator
    ```

1.7.3.1. Importing a cluster with a ManagedClusterImageRegistry CRD

Complete the following steps to import a cluster with a ManagedClusterImageRegistry CRD:

1. Create a pull secret in the namespace where you want your cluster to be imported. For these steps, it is **myNamespace**.

   ```
   $ kubectl create secret docker-registry myPullSecret \
     --docker-server=<your-registry-server> \
     --docker-username=<my-name> \
     --docker-password=<my-password>
   ```

2. Create a Placement in the namespace that you created.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: myPlacement
  namespace: myNamespace
spec:
  clusterSets:
  - myClusterSet
  tolerations:
  - key: "cluster.open-cluster-management.io/unreachable"
    operator: Exists
```

Note: The **unreachable** toleration is required for the Placement to be able to select the cluster.

3. Create a **ManagedClusterSet** resource and bind it to your namespace.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSet
metadata:
  name: myClusterSet

---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSetBinding
metadata:
  name: myClusterSet
  namespace: myNamespace
spec:
  clusterSet: myClusterSet
```

4. Create the **ManagedClusterImageRegistry** CRD in your namespace.

```
apiVersion: imageregistry.open-cluster-management.io/v1alpha1
kind: ManagedClusterImageRegistry
metadata:
  name: myImageRegistry
  namespace: myNamespace
spec:
  placementRef:
    group: cluster.open-cluster-management.io
    resource: placements
    name: myPlacement
  pullSecret:
    name: myPullSecret
  registry: myRegistryAddress
```

5. Import a managed cluster from the Red Hat Advanced Cluster Management console and add it to a managed cluster set.

6. Copy and run the import commands on the managed cluster after the label **open-cluster-management.io/image-registry=myNamespace.myImageRegistry** is added to the managed cluster.

1.7.4. Modifying the klusterlet add-ons settings of your cluster

You can modify the settings of **KlusterletAddonConfig** to change your configuration using the hub cluster.

The **KlusterletAddonConfig** controller manages the functions that are enabled and disabled according to the settings in the **klusterletaddonconfigs.agent.open-cluster-management.io** Kubernetes resource. View the following example of the **KlusterletAddonConfig**:

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster-name>
  namespace: <cluster-name>
spec:
  clusterName: <cluster-name>
  clusterNamespace: <cluster-name>
  clusterLabels:
    cloud: auto-detect
    vendor: auto-detect
  applicationManager:
    enabled: true
  certPolicyController:
    enabled: true
  iamPolicyController:
    enabled: true
  policyController:
    enabled: true
  searchCollector:
    enabled: false
  version: 2.5.0
```

1.7.4.1. Description of klusterlet add-ons settings

The following settings can be updated in the **klusterletaddonconfigs.agent.open-cluster-management.io** Kubernetes resource:

Table 1.2. Table list of klusterlet add-ons settings

Setting name	Value	Description
applicationmanager	**true** or **false**	This controller manages the application subscription lifecycle on the managed cluster.
certPolicyController	**true** or **false**	This controller enforces certificate-based policies on the managed cluster.
iamPolicyController	**true** or **false**	This controller enforces the IAM-based policy lifecycle on the managed cluster.

Setting name	Value	Description
policyController	**true** or **false**	This controller enforces all other policy rules on the managed cluster.
searchCollector	**true** or **false**	This controller is used to periodically push the resource index data back to the hub cluster.

1.7.4.2. Modify using the console on the hub cluster

You can modify the settings of the **klusterletaddonconfigs.agent.open-cluster-management.io** resource by using the hub cluster. Complete the following steps to change the settings:

1. Log in to the Red Hat Advanced Cluster Management for Kubernetes console of the hub cluster.
2. From the header menu of the hub cluster console, select the **Search** icon.
3. In the search parameters, enter the following value: **kind:klusterletaddonconfigs**
4. Select the endpoint resource that you want to update.
5. Find the **spec** section and select **Edit** to edit the content.
6. Modify your settings.
7. Select **Save** to apply your changes.

1.7.4.3. Modify using the command line on the hub cluster

You must have access to the **cluster-name** namespace to modify your settings by using the hub cluster. Complete the following steps:

1. Log in to the hub cluster.
2. Enter the following command to edit the resource:

```
kubectl edit klusterletaddonconfigs.agent.open-cluster-management.io <cluster-name> -n <cluster-name>
```

3. Find the **spec** section.
4. Modify your settings, as necessary.

1.8. ACCESSING YOUR CLUSTER

To access an Red Hat OpenShift Container Platform cluster that was created and is managed by Red Hat Advanced Cluster Management for Kubernetes, complete the following steps:

1. From the Red Hat Advanced Cluster Management for Kubernetes navigation menu, navigate to **Infrastructure** > **Clusters** and select the name of the cluster that you created or want to access.

2. Select **Reveal credentials** to view the user name and password for the cluster. Note these values to use when you log in to the cluster.
   Note: The **Reveal credentials** option is not available for imported clusters.

3. Select **Console URL** to link to the cluster.

4. Log in to the cluster by using the user ID and password that you found in step three.

1.9. CREATING A CLUSTER IN A PROXY ENVIRONMENT

You can create a Red Hat OpenShift Container Platform cluster when your hub cluster is connected through a proxy server.

One of the following situations must be true for the cluster creation to succeed:

- Red Hat Advanced Cluster Management for Kubernetes has a private network connection with the managed cluster that you are creating, but the Red Hat Advanced Cluster Management and managed cluster access the Internet using a proxy.

- The managed cluster is on a infrastructure provider, but the firewall ports enable communication from the managed cluster to the hub cluster.

To create a cluster that is configured with a proxy, complete the following steps:

1. Configure the cluster-wide-proxy setting on the hub cluster by adding the following information to your **install-config.yaml** file:

   ```
   apiVersion: v1
   kind: Proxy
   baseDomain: <domain>
   proxy:
     httpProxy: http://<username>:<password>@<proxy.example.com>:<port>
     httpsProxy: https://<username>:<password>@<proxy.example.com>:<port>
     noProxy: <wildcard-of-domain>,<provisioning-network/CIDR>,<BMC-address-range/CIDR>
   ```

   Replace **username** with the username for your proxy server.

   Replace **password** with the password to access your proxy server.

   Replace **proxy.example.com** with the path of your proxy server.

   Replace **port** with the communication port with the proxy server.

   Replace **wildcard-of-domain** with an entry for domains that should bypass the proxy.

   Replace **provisioning-network/CIDR** with the IP address of the provisioning network and the number of assigned IP addresses, in CIDR notation.

   Replace **BMC-address-range/CIDR** with the BMC address and the number of addresses, in CIDR notation.

   After you add the previous values, the settings are applied to your clusters.

2. Provision the cluster by completing the procedure for creating a cluster. See Creating a cluster to select your provider.

1.9.1. Enabling cluster-wide proxy on existing cluster add-ons

You can configure the **KlusterletAddonConfig** in the cluster namespace to add the proxy environment variables to all of the klusterlet add-on pods of the Red Hat OpenShift Container Platform clusters that are managed by the hub cluster.

Complete the following steps to configure the **KlusterletAddonConfig** to add the 3 environment variables to the pods of the klusterlet add-ons:

1. Open the **KlusterletAddonConfig** file that is in the namespace of the cluster that needs the proxy added.

2. Edit the **.spec.proxyConfig** section of the file so it resembles the following example:

```
spec
  proxyConfig:
    httpProxy: "<proxy_not_secure>"
    httpsProxy: "<proxy_secure>"
    noProxy: "<no_proxy>"
```

Replace **proxy_not_secure** with the address of the proxy server for **http** requests. For example, **http://192.168.123.145:3128**.

Replace **proxy_secure** with the address of the proxy server for **https** requests. For example, **https://192.168.123.145:3128**.

Replace **no_proxy** with a comma delimited list of IP addresses, hostnames, and domain names where traffic will not be routed through the proxy. For example, **.cluster.local,.svc,10.128.0.0/14,example.com**.

The **spec.proxyConfig** is an optional section. If the OpenShift Container Platform cluster is created with cluster wide proxy configured on the Red Hat Advanced Cluster Management hub cluster, the cluster wide proxy configuration values are added to the pods of the klusterlet add-ons as environment variables when the following conditions are met:

- The **.spec.policyController.proxyPolicy** in the **addon** section is enabled and set to **OCPGlobalProxy**

- The **.spec.applicationManager.proxyPolocy** is enabled and set to **CustomProxy**. **Note:** The default value of **proxyPolicy** in the **addon** section is **Disabled**.

  See the following example:

```
apiVersion: agent.open-cluster-management.io/v1
   kind: KlusterletAddonConfig
   metadata:
     name: clusterName
     namespace: clusterName
   spec:
     proxyConfig:
       httpProxy: http://pxuser:12345@10.0.81.15:3128
       httpsProxy: http://pxuser:12345@10.0.81.15:3128
       noProxy: .cluster.local,.svc,10.128.0.0/14, example.com
```

```
  applicationManager:
    enabled: true
    proxyPolicy: CustomProxy
  policyController:
    enabled: true
    proxyPolicy: OCPGlobalProxy
  searchCollector:
    enabled: true
    proxyPolicy: Disabled
  certPolicyController:
    enabled: true
    proxyPolicy: Disabled
  iamPolicyController:
    enabled: true
    proxyPolicy: Disabled
```

Proxy is configured on your cluster add-ons.

Important: Global proxy settings do not impact alert forwarding. To set up alert forwarding for Red Hat Advanced Cluster Management hub clusters with a cluster-wide proxy, see Forwarding alerts for more details.

1.10. ENABLING CLUSTER PROXY ADD-ONS

In some environments, a managed cluster is behind a firewall and cannot be accessed directly by the hub cluster. To gain access, you can set up a proxy add-on to access the **kube-api** server of the managed cluster to provide a more secure connection.

Required access: Editor

To configure a cluster proxy add-on for a hub cluster and a managed cluster, complete the following steps:

1. Enable the cluster proxy add-on on Red Hat Advanced Cluster Management for Kubernetes hub cluster. See Advanced configuration to learn more.

2. Configure the **kubeconfig** file to access the managed cluster **kube-apiserver** by completing the following steps:

   a. Provide a valid access token for the managed cluster. You can use the corresponding token of the service account, assuming the default service account is in the default namespace.

      i. Make sure that you are using the context of the managed cluster. Assume the file named **managed-cluster.kubeconfig** is the **kubeconfig** file of the managed cluster. **Tip:** Commands with **--kubeconfig=managed-cluster.kubeconfig** run on the managed cluster, and all of the commands in this procedure should run in the same console. Do not run commands in different consoles.

      ii. Add a role to your service account that allows it to access pods by running the following commands:

```
oc create role -n default test-role --verb=list,get --resource=pods --
kubeconfig=managed-cluster.kubeconfig
oc create rolebinding -n default test-rolebinding --serviceaccount=default:default --
role=test-role --kubeconfig=managed-cluster.kubeconfig
```

iii. Run the following command to locate the secret of the service account token:

```
oc get secret -n default --kubeconfig=managed-cluster.kubeconfig | grep default-token
```

iv. Run the following command to copy the token:

```
export MANAGED_CLUSTER_TOKEN=$(kubectl --kubeconfig=managed-cluster.kubeconfig -n default get secret <default-token> -o jsonpath={.data.token} | base64 -d)
```

Replace **default-token** with the name of your secret.

b. Configure the **kubeconfig** file on the Red Hat Advanced Cluster Management hub cluster.

i. Export the current **kubeconfig** file on the hub cluster by running the following command:

```
oc config view --minify --raw=true > cluster-proxy.kubeconfig
```

ii. Modify the **server** file with your editor. This example uses commands when using **sed**. Run **alias sed=gsed**, if you are using OSX.

```
export TARGET_MANAGE_CLUSTER=<cluster1>

export NEW_SERVER=https://$(oc get route -n open-cluster-management cluster-proxy-addon-user -o=jsonpath='{.spec.host}')/$TARGET_MANAGE_CLUSTER

sed -i'' -e '/server:/c\    server: "'$NEW_SERVER'"' cluster-proxy.kubeconfig

export CADATA=$(oc get configmap -n openshift-service-ca kube-root-ca.crt -o=go-template='{{index .data "ca.crt"}}' | base64)

sed -i'' -e '/certificate-authority-data:/c\    certificate-authority-data: "'$CADATA'"' cluster-proxy.kubeconfig
```

Replace **cluster1** with a managed cluster name that you want to access.

iii. Delete the original user credentials by entering the following commands:

```
sed -i'' -e '/client-certificate-data/d' cluster-proxy.kubeconfig
sed -i'' -e '/client-key-data/d' cluster-proxy.kubeconfig
sed -i'' -e '/token/d' cluster-proxy.kubeconfig
```

iv. Add the token of the service account:

```
sed -i'' -e '$a\    token: "'$MANAGED_CLUSTER_TOKEN'"' cluster-proxy.kubeconfig
```

3. List all of the pods on the target namespace of the target managed cluster by running the following command:

```
oc get pods --kubeconfig=cluster-proxy.kubeconfig -n <default>
```

Replace the **default** namespace with the namespace that you want to use.

Your hub cluster is now communicating with the **kube-api** of your managed cluster.

1.11. CONFIGURING A SPECIFIC CLUSTER MANAGEMENT ROLE

When you install Red Hat Advanced Cluster Management for Kubernetes, the default configuration provides the **cluster-admin** role on the Red Hat Advanced Cluster Management hub cluster. This permission enables you to create, manage, and import managed clusters on the hub cluster. In some situations, you might want to limit the access to certain managed clusters that are managed by the hub cluster, rather than providing access to all of the managed clusters on the hub cluster.

You can limit access to certain managed clusters by defining a cluster role and applying it to a user or group. Complete the following steps to configure and apply a role:

1. Define the cluster role by creating a YAML file with the following content:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <clusterrole-name>
rules:
- apiGroups:
  - cluster.open-cluster-management.io
  resources:
  - managedclusters
  resourceNames:
  - <managed-cluster-name>
  verbs:
  - get
  - list
  - watch
  - update
  - delete
  - deletecollection
  - patch
- apiGroups:
  - cluster.open-cluster-management.io
  resources:
  - managedclusters
  verbs:
  - create
- apiGroups:
  - ""
  resources:
  - namespaces
  resourceNames:
  - <managed-cluster-name>
  verbs:
  - create
  - get
  - list
  - watch
  - update
  - delete
  - deletecollection
  - patch
- apiGroups:
```

```
- register.open-cluster-management.io
resources:
- managedclusters/accept
resourceNames:
- <managed-cluster-name>
verbs:
- update
```

Replace ***clusterrole-name*** with the name of the cluster role that you are creating.

Replace ***managed-cluster-name*** with the name of the managed cluster that you want the user to have access to.

2. Apply the **clusterrole** definition by entering the following command:

```
oc apply <filename>
```

Replace ***filename*** with the name of the YAML file that you created in the previous step.

3. Enter the following command to bind the **clusterrole** to a specified user or group:

```
oc adm policy add-cluster-role-to-user <clusterrole-name> <username>
```

Replace ***clusterrole-name*** with the name of the cluster role that you applied in the previous step. Replace ***username*** with the username to which you want to bind the cluster role.

1.12. MANAGING CLUSTER LABELS

Add a label to your cluster to select group resources. See Labels and Selectors for more information.

You can add new labels, remove existing labels, and edit existing labels for your clusters.

To manage your labels, navigate to **Infrastructure** > **Clusters** and find your cluster in the *Clusters* table. Use the **Options** menu for the cluster to select **Edit labels**.

- To add a new label, enter a label in the *Edit labels* dialog box. Your entry must be in the following format: **Key=Value**. If you are adding multiple labels, separate the labels by pressing **enter**, adding a comma, or adding a space between the labels.
  Labels are only saved after you click **Save**.

- To remove an existing label, click the **Remove** icon for the label that you want to remove in the list.

- To update an existing label, you can reassign its key to a new value by adding a new label using the same key with a different value. For example, you can change **Key=Value** by entering **Key=NewValue** to update the value of **Key**.

Tip: You can also edit a cluster label from the cluster details page. In the navigation menu, click **Infrastructure** > **Clusters**. From the Clusters page, access the details page for the cluster by clicking the name of the cluster. Select the **Edit** icon in the *Labels* section. The *Edit labels* dialog box is displayed.

1.13. CONFIGURING ANSIBLE TOWER TASKS TO RUN ON MANAGED CLUSTERS

Red Hat Advanced Cluster Management is integrated with Ansible Tower automation so that you can create prehook and posthook AnsibleJob instances that occur before or after creating or upgrading your clusters. Configuring prehook and posthook jobs for cluster destroy, and cluster scale actions are not supported.

Required access: Cluster administrator

- Prerequisites
- Configuring an AnsibleJob template to run on a cluster by using the console
- Creating an AnsibleJob template
- Configuring an AnsibleJob template to run on a managed cluster by using labels
- Viewing the status of an Ansible job

1.13.1. Prerequisites

You must meet the following prerequisites to run Ansible templates on your Red Hat Advanced Cluster Management clusters:

- OpenShift Container Platform 4.6 or later
- Install the Ansible Automation Platform Resource Operator to connect Ansible jobs to the lifecycle of Git subscriptions. For best results when using the AnsibleJob to launch Ansible Tower jobs, the Ansible Tower job template should be idempotent when it is run. You can find the Ansible Automation Platform Resource Operator in the OpenShift Container Platform *OperatorHub*.

For more information about installing and configuring Ansible Tower automation, see Setting up Ansible tasks.

1.13.2. Configuring an *AnsibleJob* template to run on a cluster by using the console

You must specify the Ansible job template that you want to use for a cluster when you create the cluster. To specify the template when creating a cluster, select the Ansible template that you want to apply to the cluster in the *Automation* step. If there are no Ansible templates, click **Add automation template** to create one.

1.13.3. Creating an *AnsibleJob* template

To initiate an Ansible job with a cluster installation or upgrade, you must create an Ansible job template to specify when you want the jobs to run. They can be configured to run before or after the cluster installs or upgrades.

To specify the details about running the Ansible template while creating a template, complete the steps in the console:

1. Select **Infrastructure** > **Automation** from the Red Hat Advanced Cluster Management navigation.
2. Select the applicable path for your situation:
   - If you want to create a new template, click **Create Ansible template** and continue with step 3.

- If you want to modify an existing template, click **Edit template** from the *Options* menu of the template that you want to modify and continue with step 5.

3. Enter a unique name for your template, which contains lowercase alphanumeric characters or a hyphen (-).

4. Select the credential that you want to use for the new template. To link an Ansible credential to an Ansible template, complete the following steps:

   a. From the Red Hat Advanced Cluster Management navigation, select **Automation**. Any template in the list of templates that is not linked to a credential contains a **Link to credential** icon that you can use to link the template to an existing credential. Only the credentials in the same namespace as the template are displayed.

   b. If there are no credentials that you can select, or if you do not want to use an existing credential, select **Edit template** from the *Options* menu for the template that you want to link.

   c. Click **Add credential** and complete the procedure in Creating a credential for Ansible Automation Platform if you have to create your credential.

   d. After you create your credential in the same namespace as the template, select the credential in the *Ansible Automation Platform credential* field when you edit the template.

5. If you want to initiate any Ansible jobs before the cluster is installed, select **Add an Ansible job template** in the *Pre-install Ansible job templates* section.

6. Select or enter the name of the prehook and posthook Ansible jobs to add to the installation or upgrade of the cluster.
   Note: The *Ansible job template name* must match the name of the Ansible job on the Ansible Tower.

7. Drag the Ansible jobs to change the order, if necessary.

8. Repeat steps 5 - 7 for any Ansible job templates that you want to initiate after the cluster is installed in the *Post-install Ansible job templates* section, the *Pre-upgrade Ansible job templates* section, and the *Post-upgrade Ansible job templates* section.

Your Ansible template is configured to run on clusters that specify this template when the designated actions occur.

1.13.4. Configuring an AnsibleJob template to run on a managed cluster by using labels

You can create an **AnsibleJob** that is bound to a cluster when the cluster is created by Red Hat Advanced Cluster Management for Kubernetes or imported to be managed by Red Hat Advanced Cluster Management by using labels.

Complete the following procedure to create an Ansible job and configure it with a cluster that is not yet managed by Red Hat Advanced Cluster Management:

1. Create the definition file for the Ansible job in one of the channels that are supported by the application function. Only Git channels are supported.
   Use **AnsibleJob** as the value of **kind** in the definition.

   Your definition file contents might resemble the following example:

   ▪

```
apiVersion: apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleJob
metadata:
  name: hive-cluster-gitrepo
spec:
  tower_auth_secret: my-toweraccess
  job_template_name: my-tower-template-name
  extra_vars:
    variable1: value1
    variable2: value2
```

By storing the file in the prehook or posthook directory, it creates a list of cluster names that match the placement rule. The list of cluster names can be passed as a value of **extra_vars** to the **AnsibleJob kind** resource. When this value is passed to the **AnsibleJob** resource, the Ansible job can determine the new cluster name and use it in the automation.

2. Log on to your Red Hat Advanced Cluster Management hub cluster.

3. Using the Red Hat Advanced Cluster Management console, create an application with a Git subscription that references the channel where you stored the definition file that you just created. See Managing application resources for more information about creating an application and subscription.
When you create the subscription, specify a label that you can add to the cluster that you create or import later to connect this subscription with the cluster. This can be an existing label, like **vendor=OpenShift**, or a unique label that you create and define.

   Note: If you select a label that is already in use, the Ansible job automatically runs. It is best practice to include a resource in your application that is not part of the prehooks or posthooks.

   The default placement rule runs the job when it detects the cluster with the label that matches the label of the **AnsibleJob**. If you want the automation to run on all of your running clusters that are managed by the hub cluster, add the following content to the placement rule:

```
clusterConditions:
  - type: ManagedClusterConditionAvailable
    status: "True"
```

   You can either paste this into the YAML content of the placement rule or you can select the option to *Deploy to all online clusters and local cluster* on the *Application create* page of the Red Hat Advanced Cluster Management console.

4. Create or import your cluster by following the instructions in Creating a cluster or Importing a target managed cluster to the hub cluster, respectively.
When you create or import the cluster, use the same label that you used when you created the subscription, and the **AnsibleJob** is automatically configured to run on the cluster.

Red Hat Advanced Cluster Management automatically injects the cluster name into the **AnsibleJob.extra_vars.target_clusters** path. You can dynamically inject the cluster name into the definition. Complete the following procedure to create an AnsibleJob and configure it with a cluster that is already managed by Red Hat Advanced Cluster Management:

1. Create the definition file for the AnsibleJob in the prehook or posthook directory of your Git Channel.
Use **AnsibleJob** as the value of **kind** in the definition.

   Your definition file contents might resemble the following example:

```
apiVersion: tower.ansible.com/v1alpha1
kind: AnsibleJob
metadata:
  name: hive-cluster-gitrepo
spec:
  tower_auth_secret: my-toweraccess
  job_template_name: my-tower-template-name
  extra_vars:
    variable1: value1
    variable2: value2
```

Replace ***my-toweraccess*** with the authentication secret to access your Ansible Tower.

Replace ***my-tower-template-name*** with the template name from your Ansible Tower.

Each time a cluster that is controlled by the Ansible job is removed or added, the AnsibleJob automatically runs and updates the **extra_vars.target_clusters** variable. This updating provides the ability to specify cluster names with a specific automation, or apply the automation to a group of clusters.

1.13.5. Viewing the status of an Ansible job

You can view the status of a running Ansible job to ensure that it started, and is running successfully. To view the current status of a running Ansible job, complete the following steps:

1. In the Red Hat Advanced Cluster Management menu, select **Infrastructure** > **Clusters** to access the *Clusters* page.
2. Select the name of the cluster to view its details.
3. View the status of the last run of the Ansible job on the cluster information. The entry shows one of the following statuses:
   - When an install prehook or posthook job fails, the cluster status shows **Failed**.
   - When an upgrade prehook or posthook job fails, a warning is displayed in the *Distribution* field that the upgrade failed.
     Tip: You can retry an upgrade from the *Clusters* page if the cluster prehook or posthook failed.

1.14. CREATING AND MANAGING MANAGEDCLUSTERSETS

A **ManagedClusterSet** is a group of managed clusters. With a managed cluster set, you can manage access to all of the managed clusters in the group together. You can also create a **ManagedClusterSetBinding** resource to bind a **ManagedClusterSet** resource to a namespace.

Each managed cluster must be a member of a **ManagedClusterSet**. When you install the hub cluster, a default **ManagedClusterSet** is created called **default**. All managed clusters that are not specifically assigned to a managed cluster set are automatically assigned to the **default** managed cluster set. To ensure that the default managed cluster set is always available, you cannot delete or update the **default** managed cluster set.

Note: Cluster pools that are not specifically added to a **ManagedClusterSet** are not added to the default **ManagedClusterSet**. After a managed cluster is claimed from a cluster pool, it is added to the default **ManagedClusterSet** if it is not specifically added to another **ManagedClusterSet**.

- Creating a ManagedClusterSet
- Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet
- Creating a ManagedClusterSetBinding resource
- Adding a cluster to a ManagedClusterSet
- Removing a cluster from a ManagedClusterSet

1.14.1. Creating a ManagedClusterSet

You can group managed clusters together in a managed cluster set to limit the user access on managed clusters.

Required access: Cluster administrator

A **ManagedClusterSet** is a cluster-scoped resource, so you must have cluster administration permissions for the cluster where you are creating the **ManagedClusterSet**. A managed cluster cannot be included in more than one **ManagedClusterSet**. You can create a managed cluster set from either the Red Hat Advanced Cluster Management for Kubernetes console or from the command-line interface.

1.14.1.1. Creating a ManagedClusterSet by using the console

Complete the following steps to create a managed cluster set by using the Red Hat Advanced Cluster Management console:

1. In the main console navigation, select **Infrastructure** > **Clusters** and ensure that the *Cluster sets* tab is selected.
2. Select **Create cluster set**, and enter the name of the cluster set.

1.14.1.2. Creating a ManagedClusterSet by using the command line

Add the following definition of the managed cluster set to your **yaml** file to create a managed cluster set by using the command line:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSet
metadata:
  name: <clusterset1>
```

Replace **clusterset1** with the name of your managed cluster set.

1.14.2. Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet

You can assign users or groups to your cluster set that are provided by the configured identity providers on the hub cluster.

Required access: Cluster administrator

The **ManagedClusterSet** API offers two levels of RBAC permissions:

- Cluster set **admin**
  - Full access permissions to all of the cluster and cluster pool resources that are assigned to the managed cluster set.
  - Permission to create clusters, import clusters, and create cluster pools. The permissions must be assigned to the managed cluster set when the managed cluster set is created.
- Cluster set **view**
  - Read only permissions to all of the cluster and cluster pool resources that are assigned to the managed cluster set.
  - No permission to create clusters, import clusters, or create cluster pools.

Complete the following steps to assign users or groups to your managed cluster set from the Red Hat Advanced Cluster Management console:

1. From the main navigation menu of the console, select **Infrastructure** > **Clusters**.
2. Select the *Cluster sets* tab.
3. Select your target cluster set.
4. Select the *Access management* tab.
5. Select **Add user or group**.
6. Search for, and select the user or group that you want to provide access.
7. Select the **Cluster set admin** or **Cluster set view** role to give to the selected user or user group. See Overview of roles for more information about the role permissions.
8. Select **Add** to submit the changes.

Your user or group is displayed in the table. It might take a few seconds for the permission assignments for all of the managed cluster set resources to be propagated to your user or group.

For more information about role-based actions, see Role-based access control.

See Using ManagedClusterSets with Placement for placement information.

1.14.2.1. Creating a ManagedClusterSetBinding resource

Create a **ManagedClusterSetBinding** resource to bind a **ManagedClusterSet** resource to a namespace. Applications and policies that are created in the same namespace can only access managed clusters that are included in the bound managed cluster set resource.

Access permissions to the namespace automatically apply to a managed cluster set that is bound to that namespace. If you have access permissions to access the namespace to which the managed cluster set is bound, you automatically have permissions to access any managed cluster set that is bound to that namespace. However, if you only have permissions to access the managed cluster set, you do not automatically have permissions to access other managed cluster sets on the namespace. If you do not see a managed cluster set, you might not have the required permissions to view it.

You can create a managed cluster set binding by using the console or the command line.

1.14.2.1.1. Creating a ManagedClusterSetBinding by using the console

Complete the following steps to remove a cluster from a managed cluster set by using the Red Hat Advanced Cluster Management console:

1. Access the cluster page by selecting **Infrastructure** > **Clusters** in the main navigation and select the *Cluster sets* tab.

2. Select the name of the cluster set that you want to create a binding for to view the cluster set details.

3. Select **Actions** > **Edit namespace bindings**.

4. On the *Edit namespace bindings* page, select the namespace to which you want to bind the cluster set from the drop-down menu. The existing namespaces that have bindings to the cluster set are already selected.

1.14.2.1.2. Creating a ManagedClusterSetBinding by using the command line

To create a managed cluster set binding by using the command line, complete the following steps:

1. Create the **ManagedClusterSetBinding** resource in your **yaml** file. When you create a managed cluster set binding, the name of the managed cluster set binding must match the name of the managed cluster set to bind. Your **ManagedClusterSetBinding** resource might resemble the following information:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSetBinding
metadata:
  namespace: project1
  name: clusterset1
spec:
  clusterSet: clusterset1
```

2. Ensure that you have the bind permission on the target managed cluster set. View the following example of a **ClusterRole** resource, which contains rules that allow the user to bind to **clusterset1**:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: clusterrole1
rules:
  - apiGroups: ["cluster.open-cluster-management.io"]
    resources: ["managedclustersets/bind"]
    resourceNames: ["clusterset1"]
    verbs: ["create"]
```

1.14.3. Adding a cluster to a ManagedClusterSet

After you create your **ManagedClusterSet**, you must add one or more managed clusters. You can add managed clusters to your managed cluster set by using either the console or the command line.

1.14.3.1. Adding clusters to a ManagedClusterSet by using the console

Complete the following steps to add a cluster to a managed cluster set by using the Red Hat Advanced Cluster Management console:

1. If you just created your managed cluster set, select **Manage resource assignments** to go directly to the *Manage resource assignments* page. Continue with step 6 of this procedure.
2. If your cluster already exists, access the cluster page by selecting **Infrastructure** > **Clusters** in the main navigation.
3. Select the **Cluster sets** tab to view the available cluster sets.
4. Select the name of the cluster set that you want to add to the managed cluster sets to view the cluster set details.
5. Select **Actions** > **Manage resource assignments**.
6. On the *Manage resource assignments* page, select the checkbox for the resources that you want to add to the cluster set.
7. Select **Review** to review your changes.
8. Select **Save** to save your changes.
   Note: If you move a managed cluster from one managed cluster set to another, you must have the required RBAC permission available on both managed cluster sets.

1.14.3.2. Adding clusters to a ManagedClusterSet by using the command line

Complete the following steps to add a cluster to a managed cluster set by using the command line:

1. Ensure that there is an RBAC **ClusterRole** entry that allows you to create on a virtual subresource of **managedclustersets/join**. Without this permission, you cannot assign a managed cluster to a **ManagedClusterSet**.
   If this entry does not exist, add it to your **yaml** file. A sample entry resembles the following content:

   ```
   kind: ClusterRole
   apiVersion: rbac.authorization.k8s.io/v1
   metadata:
     name: clusterrole1
   rules:
     - apiGroups: ["cluster.open-cluster-management.io"]
       resources: ["managedclustersets/join"]
       resourceNames: ["<clusterset1>"]
       verbs: ["create"]
   ```

   Replace **clusterset1** with the name of your **ManagedClusterSet**.

   Note: If you are moving a managed cluster from one **ManagedClusterSet** to another, you must have that permission available on both managed cluster sets.

2. Find the definition of the managed cluster in the **yaml** file. The section of the managed cluster definition where you add a label resembles the following content:

   ```
   apiVersion: cluster.open-cluster-management.io/v1
   kind: ManagedCluster
   metadata:
   ```

```
  name: cluster1
spec:
  hubAcceptsClient: true
```

In this example, **cluster1** is the name of the managed cluster.

3. Add a label that specifies the name of the **ManagedClusterSet** in the format: **cluster.open-cluster-management.io/clusterset: clusterset1**.
Your code resembles the following example:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
  labels:
    cluster.open-cluster-management.io/clusterset: clusterset1
spec:
  hubAcceptsClient: true
```

In this example, **cluster1** is the cluster that is added to the managed cluster set names **clusterset1**.

Note: If the managed cluster was previously assigned to a managed cluster set that was deleted, the managed cluster might have a managed cluster set already specified to a cluster set that does not exist. If so, replace the name with the new one.

1.14.4. Removing a managed cluster from a ManagedClusterSet

You might want to remove a managed cluster from a managed cluster set to move it to a different managed cluster set, or remove it from the management settings of the set. You can remove a managed cluster from a managed cluster set by using the console or the command-line interface.

Note: Every managed cluster must be assigned to a managed cluster set. If you remove a managed cluster from a **ManagedClusterSet** and do not assign it to a different **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

1.14.4.1. Removing a managed cluster from a ManagedClusterSet by using the console

Complete the following steps to remove a cluster from a managed cluster set by using the Red Hat Advanced Cluster Management console:

1. If you just created your managed cluster set, select **Manage resource assignments** to go directly to the *Manage resource assignments* page. Continue with step 5 of this procedure.

2. If your cluster already exists, access the cluster page by selecting **Infrastructure** > **Clusters** in the main navigation and ensure that the *Cluster sets* tab is selected.

3. Select the name of the cluster set that you want to remove from the managed cluster set to view the cluster set details.

4. Select **Actions** > **Manage resource assignments**.

5. On the *Manage resource assignments* page, select the checkbox for the resources that you want to remove from the cluster set.

This step removes a resource that is already a member of the cluster set, or adds a resource that is not already a member of the cluster set. You can see if the resource is already a member of a cluster set by viewing the details of the managed cluster.

Note: If you are moving a managed cluster from one managed cluster set to another, you must have the required RBAC permissions on both managed cluster sets.

1.14.4.2. Removing clusters from a ManagedClusterSet by using the command line

To remove a managed cluster from a managed cluster set by using the command line, complete the following steps:

1. Run the following command to display a list of managed clusters in the managed cluster set:

```
oc get managedclusters -l cluster.open-cluster-management.io/clusterset=<clusterset1>
```

   Replace **clusterset1** with the name of the managed cluster set.

2. Locate the entry for the cluster that you want to remove.

3. Remove the label from the the **yaml** entry for the cluster that you want to remove. See the following code for an example of the label:

```
labels:
  cluster.open-cluster-management.io/clusterset: clusterset1
```

Note: If you are moving a managed cluster from one managed cluster set to another, you must have the required RBAC permission available on both managed cluster sets.

1.14.5. Using ManagedClusterSets with Placement

A **Placement** resource is a namespace-scoped resource that defines a rule to select a set of **ManagedClusters** from the **ManagedClusterSets**, which are bound to the placement namespace.

Required access: Cluster administrator, Cluster set administrator

1.14.5.1. Placement overview

See the following information about how placement with managed clusters works:

- Kubernetes clusters are registered with the hub cluster as cluster-scoped **ManagedClusters**.
- The **ManagedClusters** are organized into cluster-scoped **ManagedClusterSets**.
- The **ManagedClusterSets** are bound to workload namespaces.
- The namespace-scoped **Placements** specify a portion of **ManagedClusterSets** that select a working set of the potential **ManagedClusters**.
- **Placements** select from that working set by using label and claim selectors.
  Important: Placement does not select a **ManagedCluster** if there is no **ManagedClusterSet** bound to the placement namespace.
- The placement of **ManagedClusters** can be controlled by using taints and tolerations. See Using taints and tolerations to place managed clusters for more information.

The **Placement** specification includes the following fields:

- **ClusterSets** represents the **ManagedClusterSets** from which the **ManagedClusters** are selected.
  - If not specified, **ManagedClusters** is selected from the **ManagedClusterSets** that are bound to the placement namespace.
  - If specified, **ManagedClusters** is selected from the intersection of this set and the **ManagedClusterSets** that are bound to the placement namespace.
- **NumberOfClusters** represents the desired number of **ManagedClusters** to be selected, which meet the placement requirements.
  If not specified, all **ManagedClusters** that meet the placement requirements are selected.
- **Predicates** represents a slice of predicates to select **ManagedClusters** with label and claim selector. The predicates are ORed.
- **prioritizerPolicy** represents the policy of prioritizers.
  - **mode** is either **Exact**, **Additive**, or **""** where **""** is **Additive** by default.
    - In **Additive** mode, any prioritizer that is not specifically provided with configuration values is enabled with its default configurations. In the current default configuration, the *Steady* and *Balance* prioritizers have the weight of 1, while the other prioritizers have the weight of 0. The default configurations might change in the future, which might change the prioritization. **Additive** mode does not require that you configure all of the prioritizers.
    - In **Exact** mode, any prioritizer that is not specifically provided with configuration values has the weight of zero. **Exact** mode requires that you input the full set of prioritizers that you want, but avoids behavior changes between releases.
  - **configurations** represents the configuration of prioritizers.
    - scoreCoordinate represents the configuration of the prioritizer and score source.
      - **type** defines the type of the prioritizer score. Type is either **BuiltIn**, **AddOn**, or **" "**, where **" "** is **BuiltIn** by default. When the type is **BuiltIn**, the built in prioritizer name must be specified. When the type is **AddOn**, you need to configure the score source in **AddOn**.
      - **builtIn** defines the name of a BuiltIn prioritizer. The following list includes the valid **BuiltIn** prioritizer names:
        - Balance: Balance the decisions among the clusters.
        - Steady: Ensure the existing decision is stabilized.
        - ResourceAllocatableCPU & ResourceAllocatableMemory: Sort clusters based on the allocatable resources.
      - **addOn** defines the resource name and score name. **AddOnPlacementScore** is introduced to describe addon scores. See Extensible scheduling to learn more about it.

- **resourceName** defines the resource name of the **AddOnPlacementScore**. The placement prioritizer selects the **AddOnPlacementScore** custom resource by this name.

- **scoreName** defines the score name inside of the **AddOnPlacementScore**. **AddOnPlacementScore** contains a list of the score name and the score value. The **scoreName**, specifies the score to be used by the prioritizer.

- **weight** defines the weight of prioritizer. The value must be in the range of [-10,10]. Each prioritizer calculates an integer score of a cluster in the range of [-100, 100]. The final score of a cluster is determined by the following formula **sum(weight * prioritizer_score)**. A higher weight indicates that the prioritizer receives a higher weight in the cluster selection, while 0 weight indicates that the prioritizer is disabled. A negative weight indicates that it is one of the last ones selected.

Note: The **configurations.name** file will be removed in v1beta1 and replaced by the **scoreCoordinate.builtIn** file. If both **name** and **scoreCoordinate.builtIn** are defined, the value in **scoreCoordinate.builtIn** is used to determine the selection.

1.14.5.2. Placement examples

You need to bind at least one **ManagedClusterSet** to a namespace by creating a **ManagedClusterSetBinding** in that namespace. **Note:** You need role-based access to **CREATE** on the virtual sub-resource of **managedclustersets/bind**. See the following examples:

- You can select **ManagedClusters** with the **labelSelector**. See the following sample where the **labelSelector** only matches clusters with label **vendor: OpenShift**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
```

- You can select **ManagedClusters** with **claimSelector**. See the following sample where **claimSelector** only matches clusters with **region.open-cluster-management.io** with **us-west-1**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement2
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
```

```
        operator: In
        values:
          - us-west-1
```

- You can select **ManagedClusters** from particular **clusterSets**. See the following sample where **claimSelector** only matches **clusterSets: clusterset1 clusterset2**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement3
  namespace: ns1
spec:
  clusterSets:
    - clusterset1
    - clusterset2
  predicates:
    - requiredClusterSelector:
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

- Select desired number of **ManagedClusters**. See the following sample where **numberOfClusters** is **3**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement4
  namespace: ns1
spec:
  numberOfClusters: 3
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

- Select a cluster with the largest allocatable memory.
  Note: Similar to Kubernetes Node Allocatable, 'allocatable' is defined as the amount of compute resources that are available for pods on each cluster.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement6
```

```
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
```

- Select a cluster with the largest allocatable CPU and memory, and make placement sensitive to resource changes.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement7
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableCPU
        weight: 2
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
        weight: 2
```

- Select two clusters with the largest allocatable memory and the largest add-on score cpu ratio, and pin the placement decisions.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement8
  namespace: ns1
spec:
  numberOfClusters: 2
  prioritizerPolicy:
    mode: Exact
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
      - scoreCoordinate:
          builtIn: Steady
        weight: 3
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: cpuratio
```

1.14.5.3. Placement decision

One or multiple **PlacementDecisions** with label **cluster.open-cluster-management.io/placement={placement name}** are created to represent the **ManagedClusters** selected by a **Placement**.

If a **ManagedCluster** is selected and added to a **PlacementDecision**, components that consume this **Placement** might apply the workload on this **ManagedCluster**. After the **ManagedCluster** is no longer selected and it is removed from the **PlacementDecisions**, the workload that is applied on this **ManagedCluster** should be removed accordingly.

See the following **PlacementDecision** sample:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/placement: placement1
  name: placement1-kbc7q
  namespace: ns1
  ownerReferences:
    - apiVersion: cluster.open-cluster-management.io/v1beta1
      blockOwnerDeletion: true
      controller: true
      kind: Placement
      name: placement1
      uid: 05441cf6-2543-4ecc-8389-1079b42fe63e
status:
  decisions:
    - clusterName: cluster1
      reason: ''
    - clusterName: cluster2
      reason: ''
    - clusterName: cluster3
      reason: ''
```

1.14.5.4. Add-on status

You might want to select managed clusters for your placements according to the status of the add-ons that are deployed on them. For example, you want to select a managed cluster for your placement only if there is a specific add-on that is enabled on the cluster.

You can do this by specifying the label for the add-on, as well as its status, if necessary, when you create the Placement. A label is automatically created on a **ManagedCluster** resource if an add-on is enabled on the cluster. The label is automatically removed if the add-on is disabled.

Each add-on is represented by a label in the format of **feature.open-cluster-management.io/addon-<addon_name>=<status_of_addon>**.

Replace **addon_name** with the name of the add-on that should be enabled on the managed cluster that you want to select.

Replace **status_of_addon** with the status that the add-on should have if the cluster is selected. The possible values of **status_of_addon** are in the following list:

- **available**: The add-on is enabled and available.
- **unhealthy**: The add-on is enabled, but the lease is not updated continuously.
- **unreachable**: The add-on is enabled, but there is no lease found for it. This can also be caused when the managed cluster is offline.

For example, an available **application-manager** add-on is represented by a label on the managed cluster that reads:

```
feature.open-cluster-management.io/addon-application-manager: available
```

See the following examples of creating placements based on add-ons and their status:

- You can create a placement that includes all managed clusters that have **application-manager** enabled on them by adding the following YAML content:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchExpressions:
            - key: feature.open-cluster-management.io/addon-application-manager
              operator: Exists
```

- You can create a placement that includes all managed clusters that have **application-manager** enabled with an **available** status by adding the following YAML content:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement2
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            "feature.open-cluster-management.io/addon-application-manager": "available"
```

- You can create a placement that includes all managed clusters that have **application-manager** disabled by adding the following YAML content:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement3
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchExpressions:
            - key: feature.open-cluster-management.io/addon-application-manager
              operator: DoesNotExist
```

1.14.5.5. Extensible scheduling

In placement resource-based scheduling, sometimes the prioritizer needs more data than the default value provided by the **MananagedCluster** resource to calculate the score of the managed cluster. For example, schedule the clusters based on CPU or memory usage data of the clusters that are fetched obtained through a monitoring system.

The API **AddOnPlacementScore** supports a more extensible way to schedule based on customized scores.

- You can specify the score in the **placement.yaml** file to select clusters.
- As a score provider, a 3rd party controller can run on either the hub cluster or the managed cluster, to maintain the lifecycle of **AddOnPlacementScore** and update score into it.

Refer to placement extensible scheduling enhancement in the **open-cluster-management** repository to learn more.

1.14.6. Using taints and tolerations to place managed clusters

You can control the placement of your managed clusters or managed cluster sets by using taints and tolerations. Taints and tolerations provide a way to prevent managed clusters from being selected for certain placements. This control can be helpful if you want to prevent certain managed clusters from being included in some placements. You can add a taint to the managed cluster, and add a toleration to the placement. If the taint and the toleration do not match, then the managed cluster is not selected for that placement.

1.14.6.1. Adding a taint to a managed cluster

Taints are specified in the properties of a managed cluster and allow a placement to repel a managed cluster or a set of managed clusters. You can add a taint to a managed cluster by entering a command that resembles the following example:

```
kubectl taint ManagedCluster <managed_cluster_name> key=value:NoSelect
```

The specification of a taint includes the following fields:

- **Required** Key - The taint key that is applied to a cluster. This value must match the value in the toleration for the managed cluster to meet the criteria for being added to that placement. You can determine this value. For example, this value could be **bar** or **foo.example.com/bar**.
- **Optional** Value - The taint value for the taint key. This value must match the value in the toleration for the managed cluster to meet the criteria for being added to that placement. For example, this value could be **value**.
- **Required** Effect - The effect of the taint on placements that do not tolerate the taint, or what occurs when the taint and the toleration of the placement do not match. The value of the effects must be one of the following values:
  - **NoSelect** - Placements are not allowed to select a cluster unless they tolerate this taint. If the cluster was selected by the placement before the taint was set, the cluster is removed from the placement decision.
  - **NoSelectIfNew** - The scheduler cannot select the cluster if it is a new cluster. Placements can only select the cluster if they tolerate the taint and already have the cluster in their cluster decisions.

- **Required TimeAdded** - The time when the taint was added. This value is automatically set.

1.14.6.2. Identifying built-in taints to reflect the status of managed clusters

When a managed cluster is not accessible, you do not want the cluster added to a placement. The following taints are automatically added to managed clusters that are not accessible:

- **cluster.open-cluster-management.io/unavailable** - This taint is added to a managed cluster when the cluster has a condition of **ManagedClusterConditionAvailable** with status of **False**. The taint has the effect of **NoSelect** and an empty value to prevent an unavailable cluster from being scheduled. An example of this taint is provided in the following content:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
 name: cluster1
spec:
 hubAcceptsClient: true
 taints:
   - effect: NoSelect
     key: cluster.open-cluster-management.io/unavailable
     timeAdded: '2022-02-21T08:11:54Z'
```

- **cluster.open-cluster-management.io/unreachable** - This taint is added to a managed cluster when the status of the condition for **ManagedClusterConditionAvailable** is either **Unknown** or has no condition. The taint has effect of **NoSelect** and an empty value to prevent an unreachable cluster from being scheduled. An example of this taint is provided in the following content:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: cluster.open-cluster-management.io/unreachable
      timeAdded: '2022-02-21T08:11:06Z'
```

1.14.6.3. Adding a toleration to a placement

Tolerations are applied to placements, and allow the placements to repel managed clusters that do not have taints that match the tolerations of the placement. The specification of a toleration includes the following fields:

- **Optional** Key - The key matches the taint key to allow the placement.

- **Optional** Value - The value in the toleration must match the value of the taint for the toleration to allow the placement.

- **Optional** Operator - The operator represents the relationship between a key and a value. Valid operators are **equal** and **exists**. The default value is **equal**. A toleration matches a taint when the keys are the same, the effects are the same, and the operator is one of the following values:

- **equal** - The operator is **equal** and the values are the same in the taint and the toleration.
- **exists** - The wildcard for value, so a placement can tolerate all taints of a particular category.

- **Optional** Effect - The taint effect to match. When left empty, it matches all taint effects. The allowed values when specified are **NoSelect** or **NoSelectIfNew**.
- **Optional** TolerationSeconds - The length of time, in seconds, that the toleration tolerates the taint before moving the managed cluster to a new placement. If the effect value is not **NoSelect** or **PreferNoSelect**, this field is ignored. The default value is **nil**, which indicates that there is no time limit. The starting time of the counting of the **TolerationSeconds** is automatically listed as the **TimeAdded** value in the taint, rather than in the value of the cluster scheduled time or the **TolerationSeconds** added time.

The following example shows how to configure a toleration that tolerates clusters that have taints:

- Taint on the managed cluster for this example:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: gpu
      value: "true"
      timeAdded: '2022-02-21T08:11:06Z'
```

- Toleration on the placement that allows the taint to be tolerated

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal
```

With the example tolerations defined, **cluster1** could be selected by the placement because the **key: gpu** and **value: "true"** match.

Note: A managed cluster is not guaranteed to be placed on a placement that contains a toleration for the taint. If other placements contain the same toleration, the managed cluster might be placed on one of those placements.

1.14.6.4. Specifying a temporary toleration

The value of **TolerationSeconds** specifies the period of time that the toleration tolerates the taint. This temporary toleration can be helpful when a managed cluster is offline and you can transfer applications that are deployed on this cluster to another managed cluster for a tolerated time.

For example, the managed cluster with the following taint becomes unreachable:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: cluster.open-cluster-management.io/unreachable
      timeAdded: '2022-02-21T08:11:06Z'
```

If you define a placement with a value for **TolerationSeconds**, as in the following example, the workload transfers to another available managed cluster after 5 minutes.

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: Placement
metadata:
  name: demo4
  namespace: demo1
spec:
  tolerations:
    - key: cluster.open-cluster-management.io/unreachable
      operator: Exists
      tolerationSeconds: 300
----
```

The application is moved to another managed cluster after the managed cluster is unreachable for 5 minutes.

1.15. MANAGING CLUSTER POOLS (TECHNOLOGY PREVIEW)

Cluster pools provide rapid and cost-effective access to configured Red Hat OpenShift Container Platform clusters on-demand and at scale. Cluster pools provision a configurable and scalable number of OpenShift Container Platform clusters on Amazon Web Services, Google Cloud Platform, or Microsoft Azure that can be claimed when they are needed. They are especially useful when providing or replacing cluster environments for development, continuous integration, and production scenarios. You can specify a number of clusters to keep running so that they are available to be claimed immediately, while the remainder of the clusters will be kept in a hibernating state so that they can be resumed and claimed within a few minutes.

ClusterClaim resources are used to check out clusters from cluster pools. When a cluster claim is created, the pool assigns a running cluster to it. If no running clusters are available, a hibernating cluster is resumed to provide the cluster or a new cluster is provisioned. The cluster pool automatically creates new clusters and resumes hibernating clusters to maintain the specified size and number of available running clusters in the pool.

Note: When a cluster that was claimed from the cluster pool is no longer needed and is destroyed, the resources are deleted. The cluster does not return to the cluster pool.

Required access: Administrator

- Prerequisites

- Creating a cluster pool
- Claiming clusters from cluster pools
- Scaling cluster pools
- Updating the cluster pool release image
- Destroying a cluster pool

The procedure for creating a cluster pool is similar to the procedure for creating a cluster. Clusters in a cluster pool are not created for immediate use.

1.15.1. Prerequisites

See the following prerequisites before creating a cluster pool:

- You need to deploy a Red Hat Advanced Cluster Management for Kubernetes hub cluster.
- You need Internet access for your Red Hat Advanced Cluster Management for Kubernetes hub cluster so that it can create the Kubernetes cluster on the provider environment.
- You need an AWS, GCP, or Microsoft Azure provider credential. See Managing credentials overview for more information.
- You need a configured domain in your provider environment. See your provider documentation for instructions about how to configure a domain.
- You need provider login credentials.
- You need your OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: Adding a cluster pool with this procedure configures it so it automatically imports the cluster to be managed by Red Hat Advanced Cluster Management when you claim a cluster from the pool. If you would like to create a cluster pool that does not automatically import the claimed cluster for management with the cluster claim, add the following annotation to your **clusterClaim** resource:

```
kind: ClusterClaim
metadata:
  annotations:
    cluster.open-cluster-management.io/createmanagedcluster: "false"
```

The word **"false"** must be surrounded by quotation marks to indicate that it is a string.

1.15.2. Creating a cluster pool

To create a cluster pool, select **Infrastructure** > **Clusters** in the navigation menu. The *Cluster pools* tab lists the cluster pools that you can access. Select **Create cluster pool** and complete the steps in the console.

If you do not have a infrastructure credential that you want to use for the cluster pool, you can create one by selecting **Add credential**.

You can either select an existing namespace from the list, or type the name of a new one to create one. The cluster pool does not have to be in the same namespace as the clusters.

When you create a cluster pool in a cluster set, the **namespace admin** permission is applied to all of the users with **clusterset admin** permissions for the namespace where you add the cluster pool. Similarly, the **namespace view** permission is applied to the users with **clusterset view** permissions.

You can select a cluster set name if you want the RBAC roles for your cluster pool to share the role assignments of an existing cluster set. The cluster set for the clusters in the cluster pool can only be set when you create the cluster pool. You cannot change the cluster set association for the cluster pool or for the clusters in the cluster pool after you create the cluster pool. Any cluster that you claim from the cluster pool is automatically added to the same cluster set as the cluster pool.

Note: If you do not have **cluster admin** permissions, you must select a cluster set. The request to create a cluster set is rejected with a forbidden error if you do not include the cluster set name in this situation. If no cluster sets are available for you to select, contact your cluster administrator to create a cluster set and give you **clusterset admin** permissions to it.

The cluster pool size` specifies the number of clusters that you want provisioned in your cluster pool, while the cluster pool running count specifies the number of clusters that the pool keeps running and ready to claim for immediate use.

The procedure is very similar to the procedure for creating clusters.

For specific information about the information that is required for your provider, see the following information:

- Creating a cluster on Amazon Web Services
- Creating a cluster on Google Cloud Platform
- Creating a cluster on Microsoft Azure

1.15.3. Claiming clusters from cluster pools

ClusterClaim resources are used to check out clusters from cluster pools. A claim is completed when a cluster is running and ready in the cluster pool. The cluster pool automatically creates new running and hibernated clusters in the cluster pool to maintain the requirements that are specified for the cluster pool.

Note: When a cluster that was claimed from the cluster pool is no longer needed and is destroyed, the resources are deleted. The cluster does not return to the cluster pool.

Required access: Administrator

1.15.3.1. Prerequisite

You must have the following available before claiming a cluster from a cluster pool:

A cluster pool with or without available clusters. If there are available clusters in the cluster pool, the available clusters are claimed. If there are no available clusters in the cluster pool, a cluster is created to fulfill the claim. See Creating a cluster pool for information about how to create a cluster pool.

1.15.3.2. Claim the cluster from the cluster pool

When you create a cluster claim, you request a new cluster from the cluster pool. A cluster is checked out from the pool when a cluster is available. The claimed cluster is automatically imported as one of your managed clusters, unless you disabled automatic import.

Complete the following steps to claim a cluster:

1. From the navigation menu, click **Infrastructure** > **Clusters**, and select the *Cluster pools* tab.
2. Find the name of the cluster pool you want to claim a cluster from and select **Claim cluster**.

If a cluster is available, it is claimed and immediately appears in the *Managed clusters* tab. If there are no available clusters, it might take several minutes to resume a hibernated cluster or provision a new cluster. During this time, the claim status is **pending**. Expand the cluster pool to view or delete pending claims against it.

The claimed cluster remains a member of the cluster set that it was associated with when it was in the cluster pool. You cannot change the cluster set of the claimed cluster when you claim it.

1.15.4. Scaling cluster pools

You can change the number of clusters in the cluster pool by increasing or decreasing the number of clusters in the cluster pool size.

Required access: Cluster administrator

Complete the following steps to change the number of clusters in your cluster pool:

1. From the navigation menu, click **Infrastructure** > **Clusters**.
2. Select the *Cluster pools* tab.
3. In the *Options* menu for the cluster pool that you want to change, select **Scale cluster pool**.
4. Change the value of the pool size.
5. Optionally, you can update the number of running clusters to increase or decrease the number of clusters that are immediately available when you claim them.

Your cluster pools are scaled to reflect your new values.

1.15.5. Updating the cluster pool release image

When the clusters in your cluster pool remain in hibernation for some time, the Red Hat OpenShift Container Platform release image of the clusters might become backlevel. If this happens, you can upgrade the version of the release image of the clusters that are in your cluster pool.

Required access: Edit

Complete the following steps to update the OpenShift Container Platform release image for the clusters in your cluster pool:

Note: This procedure does not update clusters from the cluster pool that are already claimed in the cluster pool. After you complete this procedure, the updates to the release images only apply to the following clusters that are related to the cluster pool:

- Clusters that are created by the cluster pool after updating the release image with this procedure.
- Clusters that are hibernating in the cluster pool. The existing hibernating clusters with the old release image are destroyed, and new clusters with the new release image replace them.

1. From the navigation menu, click **Infrastructure** > **Clusters**.
2. Select the *Cluster pools* tab.
3. Find the name of the cluster pool that you want to update in the *Cluster pools* table.
4. Click the *Options* menu for the *Cluster pools* in the table, and select **Update release image**
5. Select a new release image to use for future cluster creations from this cluster pool.

The cluster pool release image is updated.

Tip: You can update the release image for multiple cluster pools with one action by selecting the box for each of the the cluster pools and using the *Actions* menu to update the release image for the selected cluster pools.

1.15.6. Destroying a cluster pool

If you created a cluster pool and determine that you no longer need it, you can destroy the cluster pool. When you destroy a cluster pool, all of the unclaimed hibernating clusters are destroyed and their resources are released.

Required access: Cluster administrator

To destroy a cluster pool, complete the following steps:

1. From the navigation menu, click **Infrastructure** > **Clusters**.
2. Select the *Cluster pools* tab.
3. In the *Options* menu for the cluster pool that you want to delete, select **Destroy cluster pool**. Any unclaimed clusters in the cluster pool are destroyed. It might take some time for all of the resources to be deleted, and the cluster pool remains visible in the console until all of the resources are deleted.
   The namespace that contained the ClusterPool will not be deleted. Deleting the namespace will destroy any clusters that have been claimed from the ClusterPool, since the ClusterClaim resources for these clusters are created in the same namespace.

Tip: You can destroy multiple cluster pools with one action by selecting the box for each of the the cluster pools and using the *Actions* menu to destroy the selected cluster pools.

1.16. CLUSTERCLAIMS

A ***ClusterClaim*** is a cluster-scoped custom resource definition (CRD) on a managed cluster. A ClusterClaim represents a piece of information that a managed cluster claims. The following example shows a claim that is identified in the YAML file:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: id.openshift.io
spec:
  value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
```

The following table shows the defined ClusterClaims that might be on a cluster that Red Hat Advanced Cluster Management for Kubernetes manages:

Claim name	Reserved	Mutable	Description
id.k8s.io	true	false	ClusterID defined in upstream proposal
kubeversion.open-cluster-management.io	true	true	Kubernetes version
platform.open-cluster-management.io	true	false	Platform the managed cluster is running on, like AWS, GCE, and Equinix Metal
product.open-cluster-management.io	true	false	Product name, like OpenShift, Anthos, EKS and GKE
id.openshift.io	false	false	OpenShift Container Platform external ID, which is only available for an OpenShift Container Platform cluster
consoleurl.openshift.io	false	true	URL of the management console, which is only available for an OpenShift Container Platform cluster
version.openshift.io	false	true	OpenShift Container Platform version, which is only available for an OpenShift Container Platform cluster

If any of the previous claims are deleted or updated on managed cluster, they are restored or rolled back to a previous version automatically.

After the managed cluster joins the hub, the ClusterClaims that are created on a managed cluster are synchronized with the status of the **ManagedCluster** resource on the hub. A managed cluster with ClusterClaims might look similar to the following example:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  labels:
    cloud: Amazon
    clusterID: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
    installer.name: multiclusterhub
    installer.namespace: open-cluster-management
```

```
    name: cluster1
    vendor: OpenShift
  name: cluster1
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
status:
  allocatable:
    cpu: '15'
    memory: 65257Mi
  capacity:
    cpu: '18'
    memory: 72001Mi
  clusterClaims:
    - name: id.k8s.io
      value: cluster1
    - name: kubeversion.open-cluster-management.io
      value: v1.18.3+6c42de8
    - name: platform.open-cluster-management.io
      value: AWS
    - name: product.open-cluster-management.io
      value: OpenShift
    - name: id.openshift.io
      value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
    - name: consoleurl.openshift.io
      value: 'https://console-openshift-console.apps.xxxx.dev04.red-chesterfield.com'
    - name: version.openshift.io
      value: '4.5'
  conditions:
    - lastTransitionTime: '2020-10-26T07:08:49Z'
      message: Accepted by hub cluster admin
      reason: HubClusterAdminAccepted
      status: 'True'
      type: HubAcceptedManagedCluster
    - lastTransitionTime: '2020-10-26T07:09:18Z'
      message: Managed cluster joined
      reason: ManagedClusterJoined
      status: 'True'
      type: ManagedClusterJoined
    - lastTransitionTime: '2020-10-30T07:20:20Z'
      message: Managed cluster is available
      reason: ManagedClusterAvailable
      status: 'True'
      type: ManagedClusterConditionAvailable
  version:
    kubernetes: v1.18.3+6c42de8
```

1.16.1. List existing ClusterClaims

You can use the **kubectl** command to list the ClusterClaims that apply to your managed cluster. This is helpful when you want to compare your ClusterClaim to an error message.

Note: Make sure you have **list** permission on resource **clusterclaims.cluster.open-cluster-management.io**.

Run the following command to list all existing ClusterClaims that are on the managed cluster:

```
kubectl get clusterclaims.cluster.open-cluster-management.io
```

1.16.2. Create custom ClusterClaims

You can create ClusterClaims with custom names on a managed cluster, which makes it easier to identify them. The custom ClusterClaims are synchronized with the status of the **ManagedCluster** resource on the hub cluster. The following content shows an example of a definition for a customized **ClusterClaim**:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: <custom_claim_name>
spec:
  value: <custom_claim_value>
```

The max length of field **spec.value** is 1024. The **create** permission on resource **clusterclaims.cluster.open-cluster-management.io** is required to create a ClusterClaim.

1.17. USING HOSTED CONTROL PLANE CLUSTERS (TECHNOLOGY PREVIEW)

Red Hat Advanced Cluster Management for Kubernetes version 2.5 with the multicluster engine operator 2.0 can deploy Red Hat OpenShift Container Platform clusters by using two different control plane configurations. The standalone configuration uses multiple dedicated virtual machines or physical machines to host the OpenShift Container Platform control plane. You can provision hosted control planes to provision the OpenShift Container Platform control plane as pods on a hosting service cluster without the need for dedicated physical machines for each control-plane.

Note: This feature also works with the multicluster engine operator 2.0 without Red Hat Advanced Cluster Management for Kubernetes.

For Red Hat Advanced Cluster Management, Amazon Web Services is supported as a Technology Preview. You can host the control planes for your Red Hat OpenShift Container Platform version 4.10.7 and later.

The control plane is run as pods that are contained in a single namespace and is associated with the hosted control plane cluster. When OpenShift Container Platform provisions this type of hosted cluster, it provisions a worker node independent of the control plane.

See the following benefits of hosted control plane clusters:

- Saves cost by removing the need to host dedicated control plane nodes
- Introduces separation between the control plane and the workloads, which improves isolation and reduces configuration errors that can require changes
- Significantly decreases the cluster provisioning time by removing the requirement for control-plane node bootstrapping
- Supports turn-key deployments or fully customized OpenShift Container Platform provisioning

See more about using hosted control planes in the following product documentation:

- Configuring hosted control planes

- Disabling hosted control plane resources

1.17.1. Configuring hosted control planes

Configuring hosted control planes requires a hosting service cluster and a hosted cluster. By deploying the HyperShift operator on an existing cluster, you can make that cluster into a hosting service cluster and start the creation of the hosted cluster.

Hosted control planes is a Technology Preview feature, so the related components are disabled by default. Enable the feature by editing the **multiclusterengine** custom resource to set the **spec.overrides.components[?(@.name=='hypershift-preview')].enabled** to **true**.

Enter the following command to ensure that the hosted control planes feature is enabled:

```
oc patch mce multiclusterengine-sample--type=merge -p '{"spec":{"overrides":{"components":
[{"name":"hypershift-preview","enabled": true}]}}}'
```

1.17.1.1. Configuring the hosting service cluster

You can deploy hosted control planes by configuring an existing cluster to function as a hosting service cluster. The hosting service cluster is the OpenShift Container Platform cluster where the control planes are hosted, and can be the hub cluster or one of the OpenShift Container Platform managed clusters.

1.17.1.1.1. Prerequisites

You must have the following prerequisites to configure a hosting service cluster:

- Multicluster engine operator installed on at least one cluster that is managed by Red Hat OpenShift Container Platform. The multicluster engine operator is automatically installed when you install Red Hat Advanced Cluster Management version 2.5, and later, and can also be installed without Red Hat Advanced Cluster Management as an operator from the OpenShift Container Platform OperatorHub.
- If you want your Red Hat Advanced Cluster Management hub cluster to be your hosting service cluster, you must configure **local-cluster** as your hosting service cluster by completing the following steps:

  1. Create a YAML file named **import-hub.yaml** that is similar to the following example:

     ```
     apiVersion: cluster.open-cluster-management.io/v1
     kind: ManagedCluster
     metadata:
       labels:
         local-cluster: "true"
       name: local-cluster
     spec:
       hubAcceptsClient: true
       leaseDurationSeconds: 60
     ```

  2. Apply the file by entering:

     ```
     oc apply -f import-hub.yaml
     ```

  A hub cluster that is also managed by itself is designated as **local-cluster** in the list of clusters.

1.17.1.1.2. Configure the hosting service cluster

Complete the following steps on the cluster where the multicluster engine operator is installed to enable an OpenShift Container Platform managed cluster as a hosting service cluster:

1. If you plan to create and manage hosted clusters on AWS, create an OIDC S3 credentials secret named **hypershift-operator-oidc-provider-s3-credentials** for the HyperShift operator. Save the secret in the managed cluster namespace, which is the namespace of the managed cluster that is used as the hosting service cluster. If you used **local-cluster**, then create the secret in the **local-cluster** namespace
   The secret must contain 3 fields. The **bucket** field contains an S3 bucket with public access to host OIDC discovery documents for your HyperShift clusters. The **credentials** field is a reference to a file that contains the credentials of the **default** profile that can access the bucket. By default, HyperShift only uses the **default** profile to operate the **bucket**. The **region** field specifies the region of the S3 bucket.

   See Getting started in the HyperShift documentation for more information about the secret. The following example shows a sample AWS secret template:

   ```
   oc create secret generic hypershift-operator-oidc-provider-s3-credentials --from-file=credentials=$HOME/.aws/credentials --from-literal=bucket=<s3-bucket-for-hypershift> --from-literal=region=<region> -n <hypershift-hosting-service-cluster>
   ```

   Note: Disaster recovery backup for the secret is not automatically enabled. Run the following command to add the label that enables the **hypershift-operator-oidc-provider-s3-credentials** secret to be backed up for disaster recovery:

   ```
   oc label secret hypershift-operator-oidc-provider-s3-credentials -n <hypershift-hosting-service-cluster> cluster.open-cluster-management.io/backup=""
   ```

2. Install the HyperShift add-on.
   The cluster that hosts the HyperShift operator is the hosting service cluster. This step uses the **hypershift-addon** to install the HyperShift operator on a managed cluster.

   a. Create the **ManagedClusterAddon** HyperShift add-on by creating a file that resembles the following example:

   ```
   apiVersion: addon.open-cluster-management.io/v1alpha1
   kind: ManagedClusterAddOn
   metadata:
     name: hypershift-addon
     namespace: <managed-cluster-name>
   spec:
     installNamespace: open-cluster-management-agent-addon
   ```

   Replace **managed-cluster-name** with the name of the managed cluster on which you want to install the HyperShift operator. If you are installing on the Red Hat Advanced Cluster Management hub cluster, then use **local-cluster** for this value.

   b. Apply the file by running the following command:

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

   Replace **filename** with the name of the file that you created.

3. Confirm that the **hypershift-addon** is installed by running the following command:

```
oc get managedclusteraddons -n <hypershift-hosting-service-cluster> hypershift-addon
```

The output resembles the following example, when the add-on is installed:

```
NAME              AVAILABLE   DEGRADED   PROGRESSING
hypershift-addon   True
```

Your HyperShift add-on is installed and the hosting service cluster is available to manage HyperShift clusters.

1.17.1.2. Deploying a hosted cluster

After installing the HyperShift operator and enabling an existing cluster as a hosting service cluster, you can provision a HyperShift hosted cluster by creating a **HypershiftDeployment** custom resource.

1. Create a cloud provider secret as a credential using the console or a file addition. You must have permissions to create infrastructure resources for your cluster, like VPCs, subnets, and NAT gateways. The account also must correspond to the account for your guest cluster, where your workers live. See Create AWS infrastructure and IAM resources separately in the HyperShift documentation for more information about the required permissions.
The following example shows the format for AWS:

```
apiVersion: v1
metadata:
  name: my-aws-cred
  namespace: default      # Where you create HypershiftDeployment resources
type: Opaque
kind: Secret
stringData:
  ssh-publickey:         # Value
  ssh-privatekey:        # Value
  pullSecret:           # Value, required
  baseDomain:            # Value, required
  aws_secret_access_key:  # Value, required
  aws_access_key_id:      # Value, required
```

- To create this secret with the console, follow the credential creation steps by accessing **Credentials** in the navigation menu.

- To create the secret using the command line, run the following commands:

```
oc create secret generic <my-secret> -n <hypershift-deployment-namespace> --from-literal=baseDomain='your.domain.com' --from-literal=aws_access_key_id='your-aws-access-key' --from-literal=aws_secret_access_key='your-aws-secret-key' --from-literal=pullSecret='your-quay-pull-secret' --from-literal=ssh-publickey='your-ssh-publickey' --from-literal=ssh-privatekey='your-ssh-privatekey'
```

Note: Disaster recovery backup for the secret is not automatically enabled. Run the following command to add a label that enables the secret to be backed up for disaster recovery:

```
oc label secret <my-secret> -n <hypershift-deployment-namespace> cluster.open-cluster-management.io/backup=""
```

2. Create a **HypershiftDeployment** custom resource file in the cloud provider secret namespace. The **HypershiftDeployment** custom resource creates the infrastructure in the provider account, configures the infrastructure compute capacity in the created infrastructure, provisions the **nodePools** that use the hosted control plane, and creates a hosted control plane on a hosting service cluster.

   a. Create a file that contains information that resembles the following example:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: HypershiftDeployment
metadata:
  name: <cluster>
  namespace: default
spec:
  hostingCluster: <hosting-service-cluster>
  hostingNamespace: clusters
  hostedClusterSpec:
    networking:
      machineCIDR: 10.0.0.0/16    # Default
      networkType: OpenShiftSDN
      podCIDR: 10.132.0.0/14      # Default
      serviceCIDR: 172.31.0.0/16  # Default
    platform:
      type: AWS
    pullSecret:
      name: <cluster>-pull-secret    # This secret is created by the controller
    release:
      image: quay.io/openshift-release-dev/ocp-release:4.10.15-x86_64  # Default
    services:
    - service: APIServer
      servicePublishingStrategy:
        type: LoadBalancer
    - service: OAuthServer
      servicePublishingStrategy:
        type: Route
    - service: Konnectivity
      servicePublishingStrategy:
        type: Route
    - service: Ignition
      servicePublishingStrategy:
        type: Route
    sshKey: {}
  nodePools:
  - name: <cluster>
    spec:
      clusterName: <cluster>
      management:
        autoRepair: false
        replace:
          rollingUpdate:
            maxSurge: 1
            maxUnavailable: 0
          strategy: RollingUpdate
```

```
    upgradeType: Replace
  platform:
    aws:
      instanceType: m5.large
    type: AWS
  release:
    image: quay.io/openshift-release-dev/ocp-release:4.10.15-x86_64 # Default
  replicas: 2
infrastructure:
  cloudProvider:
    name: <my-secret>
  configure: True
  platform:
    aws:
      region: <region>
```

Replace **cluster** with the name of the cluster.

Replace **hosting-service-cluster** with the name of the cluster that hosts the HyperShift operator.

Replace **my-secret** with the secret to access your cloud provider.

Replace **region** with the region of your cloud provider.

b. Apply the file by entering the following command:

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

You can refer to the field definitions of the API to ensure that they are correct.

3. Check the **HypershiftDeployment** status by running the following command:

```
oc get hypershiftdeployment -n default hypershift-demo -w
```

4. After the hosted cluster is created, it is automatically imported to the hub. You can verify this by viewing the cluster list in the Red Hat Advanced Cluster Management console, or by running the following command:

```
oc get managedcluster <hypershiftDeployment.Spec.infraID>
```

1.17.1.3. Accessing a hosting service cluster

You can now access your cluster. The access secrets are stored in the **hypershift-hosting-service-cluster** namespace. This namespace is the same as the name of the hosting service cluster. Learn about the following formats secret name formats:

- **kubeconfig** secret: **<hypershiftDeployment.Spec.hostingNamespace>-<hypershiftDeployment.Name>-admin-kubeconfig** (clusters-hypershift-demo-admin-kubeconfig)
- **kubeadmin** password secret: **<hypershiftDeployment.Spec.hostingNamespace>-<hypershiftDeployment.Name>-kubeadmin-password** (clusters-hypershift-demo-kubeadmin-password)

1.17.2. Disabling the hosted control plane resources

When disabling the hosted control plane cluster feature, you must destroy the HyperShift hosted cluster and uninstall the HyperShift operator.

1.17.2.1. Destroying a HyperShift hosted cluster

To destroy a HyperShift hosted cluster, delete the **HypershiftDeployment** resource by running one of the following commands:

```
oc delete -f <HypershiftDeployment_yaml_file_name>
```

or

```
oc delete hd -n <HypershiftDeployment_namespace> <HypershiftDeployment_resource_name>
```

1.17.2.2. Uninstalling the HyperShift operator

To uninstall the HyperShift operator from a management or hosting service cluster, delete the **hypershift-addon ManagedClusterAddon** from the management cluster by running the following command:

```
oc delete managedclusteraddon -n <hypershift-management-cluster> hypershift-addon
```

1.18. DISCOVERY SERVICE INTRODUCTION

You can discover OpenShift 4 clusters that are available from OpenShift Cluster Manager. After discovery, you can import your clusters to manage. The Discovery services uses the Discover Operator for back-end and console usage.

You must have an OpenShift Cluster Manager credential. See Creating a credential for Red Hat OpenShift Cluster Manager if you need to create a credential.

Required access: Administrator

- Configure Discovery with the console
- Configure Discovery using the CLI

1.18.1. Configure Discovery with the console

Use the product console to enable Discovery.

Required access: Access to the namespace where the credential was created.

1.18.1.1. Prerequisites

- You need a credential. See Creating a credential for Red Hat OpenShift Cluster Manager to connect to OpenShift Cluster Manager.

1.18.1.2. Configure Discovery

Configure Discovery in the console to find clusters. You can create multiple **DiscoveryConfig** resources with separate credentials. Follow instructions in the console.

1.18.1.3. View discovered clusters

After you set up your credentials and discover your clusters for import, you can view them in the console.

1. Click **Clusters** > **Discovered clusters**
2. View the populated table with the following information:
   - *Name* is the display name that is designated in OpenShift Cluster Manager. If the cluster does not have a display name, a generated name based on the cluster console URL is displayed. If the console URL is missing or was modified manually in OpenShift Cluster Manager, the cluster external ID is displayed.
   - *Namespace* is the namespace where you created the credential and discovered clusters.
   - *Type* is the discovered cluster Red Hat OpenShift type.
   - *Distribution version* is the discovered cluster Red Hat OpenShift version.
   - *Infrastructure provider* is the cloud provider of the discovered cluster.
   - *Last active* is the last time the discovered cluster was active.
   - *Created* when the discovered cluster was created.
   - *Discovered* when the discovered cluster was discovered.
3. You can search for any information in the table, as well. For example, to show only *Discovered clusters* in a particular namespace, search for that namespace.
4. You can now click **Import cluster** to create managed clusters. See Import discovered clusters.

1.18.1.4. Import discovered clusters

After you discover clusters, you can import clusters that appear in the *Discovered clusters* tab of the console.

1.18.1.5. Prerequisites

You need access to the namespaces that were used to configure Discovery.

1.18.1.6. Import Discovered clusters

1. Navigate to the existing *Clusters* page and click on the *Discovered clusters* tab.
2. From the *Discovered clusters* table, find the cluster that you want to import.
3. From the options menu, choose **Import cluster**.
4. For discovered clusters, you can import manually using the documentation, or you can choose Import clusters automatically.
5. To import automatically with your credentials or Kubeconfig file, copy and paste the content.

6. Click **Import**.

1.18.2. Enable Discovery using the CLI

Enable discovery using the CLI to find clusters that are available from Red Hat OpenShift Cluster Manager.

Required access: Administrator

1.18.2.1. Prerequisites

- Create a credential to connect to Red Hat OpenShift Cluster Manager.

1.18.2.2. Discovery set up and process

Note: The **DiscoveryConfig** must be named **discovery** and must be created in the same namespace as the selected **credential**. See the following **DiscoveryConfig** sample:

```
apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveryConfig
metadata:
  name: discovery
  namespace: <NAMESPACE_NAME>
spec:
  credential: <SECRET_NAME>
  filters:
    lastActive: 7
    openshiftVersions:
    - "4.10"
    - "4.9"
    - "4.8"
```

1. Replace **SECRET_NAME** with the credential that you previously set up.
2. Replace **NAMESPACE_NAME** with the namespace of **SECRET_NAME**.
3. Enter the maximum time since last activity of your clusters (in days) to discover. For example, with **lastActive: 7**, clusters that active in the last 7 days are discovered.
4. Enter the versions of Red Hat OpenShift clusters to discover as a list of strings. **Note:** Every entry in the **openshiftVersions** list specifies an OpenShift major and minor version. For example, specifying **"4.9"** will include all patch releases for the OpenShift version **4.9**, for example **4.9.1**, **4.9.2**.

1.18.2.3. View discovered clusters

View discovered clusters by running **oc get discoveredclusters -n <namespace>** where **namespace** is the namespace where the discovery credential exists.

1.18.2.3.1. DiscoveredClusters

Objects are created by the Discovery controller. These **DiscoveredClusters** represent the clusters that are found in OpenShift Cluster Manager by using the filters and credentials that are specified in the **DiscoveryConfig discoveredclusters.discovery.open-cluster-management.io** API. The value for **name** is the cluster external ID:

```
apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveredCluster
metadata:
  name: fd51aafa-95a8-41f7-a992-6fb95eed3c8e
  namespace: <NAMESPACE_NAME>
spec:
  activity_timestamp: "2021-04-19T21:06:14Z"
  cloudProvider: vsphere
  console: https://console-openshift-console.apps.qe1-vmware-pkt.dev02.red-chesterfield.com
  creation_timestamp: "2021-04-19T16:29:53Z"
  credential:
    apiVersion: v1
    kind: Secret
    name: <SECRET_NAME>
    namespace: <NAMESPACE_NAME>
  display_name: qe1-vmware-pkt.dev02.red-chesterfield.com
  name: fd51aafa-95a8-41f7-a992-6fb95eed3c8e
  openshiftVersion: 4.10
  status: Stale
```

1.19. UPGRADING YOUR CLUSTER

After you create Red Hat OpenShift Container Platform clusters that you want to manage with Red Hat Advanced Cluster Management for Kubernetes, you can use the Red Hat Advanced Cluster Management for Kubernetes console to upgrade those clusters to the latest minor version that is available in the version channel that the managed cluster uses.

In a connected environment, the updates are automatically identified with notifications provided for each cluster that requires an upgrade in the Red Hat Advanced Cluster Management console.

Important: The process for upgrading your clusters in a disconnected environment requires some additional steps to configure and mirror the required release images. It uses the operator for Red Hat OpenShift Update Service to identify the upgrades. If you are in a disconnected environment, see Upgrading disconnected clusters for the required steps.

Notes:

To upgrade to a major version, you must verify that you meet all of the prerequisites for upgrading to that version. You must update the version channel on the managed cluster before you can upgrade the cluster with the console.

After you update the version channel on the managed cluster, the Red Hat Advanced Cluster Management for Kubernetes console displays the latest versions that are available for the upgrade.

This method of upgrading only works for OpenShift Container Platform managed clusters that are in a *Ready* state.

Important: You cannot upgrade Red Hat OpenShift Kubernetes Service managed clusters or OpenShift Container Platform managed clusters on Red Hat OpenShift Dedicated by using the Red Hat Advanced Cluster Management for Kubernetes console.

To upgrade your cluster in a connected environment, complete the following steps:

1. From the navigation menu, navigate to **Infrastructure** > **Clusters**. If an upgrade is available, it is shown in the *Distribution version* column.

2. Select the clusters in *Ready* state that you want to upgrade. A cluster must be an OpenShift Container Platform cluster to be upgraded with the console.

3. Select **Upgrade**.

4. Select the new version of each cluster.

5. Select **Upgrade**.

If your cluster upgrade fails, the Operator generally retries the upgrade a few times, stops, and reports the status of the failing component. In some cases, the upgrade process continues to cycle through attempts to complete the process. Rolling your cluster back to a previous version following a failed upgrade is not supported. Contact Red Hat support for assistance if your cluster upgrade fails.

1.19.1. Selecting a channel

You can use the Red Hat Advanced Cluster Management console to select a channel for your cluster upgrades on OpenShift Container Platform version 4.6, or later. After selecting a channel, you are automatically reminded of cluster upgrades that are available for both Errata versions (4.8.1 > 4.8.2 > 4.8.3, and so on) and release versions (4.8 > 4.9, and so on).

To select a channel for your cluster, complete the following steps:

1. From the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Clusters**.

2. Select the name of the cluster that you want to change to view the *Cluster details* page. If a different channel is available for the cluster, an edit icon is displayed in the *Channel* field.

3. Click the edit icon to modify the setting in the field.

4. Select a channel in the *New channel* field.

You can find the reminders for the available channel updates in the *Cluster details* page of the cluster.

1.19.2. Upgrading disconnected clusters

You can use Red Hat OpenShift Update Service with Red Hat Advanced Cluster Management for Kubernetes to upgrade your clusters in a disconnected environment.

In some cases, security concerns prevent clusters from being connected directly to the Internet. This makes it difficult to know when upgrades are available, and how to process those upgrades. Configuring OpenShift Update Service can help.

OpenShift Update Service is a separate operator and operand that monitors the available versions of your managed clusters in a disconnected environment, and makes them available for upgrading your clusters in a disconnected environment. After OpenShift Update Service is configured, it can perform the following actions:

1. Monitor when upgrades are available for your disconnected clusters.

2. Identify which updates are mirrored to your local site for upgrading by using the graph data file.

3. Notify you that an upgrade is available for your cluster by using the Red Hat Advanced Cluster Management console.

- Prerequisites

- Prepare your disconnected mirror registry
- Deploy the operator for OpenShift Update Service
- Build the graph data init container
- Configure certificate for the mirrored registry
- Deploy the OpenShift Update Service instance
- Deploy a policy to override the default registry (optional)
- Deploy a policy to deploy a disconnected catalog source
- Deploy a policy to change the managed cluster parameter
- Viewing available upgrades
- Selecting a channel
- Upgrading the cluster

1.19.2.1. Prerequisites

You must have the following prerequisites before you can use OpenShift Update Service to upgrade your disconnected clusters:

- A deployed Red Hat Advanced Cluster Management hub cluster that is running on Red Hat OpenShift Container Platform version 4.6 or later with restricted OLM configured. See Using Operator Lifecycle Manager on restricted networks for details about how to configure restricted OLM.
  Tip: Make a note of the catalog source image when you configure restricted OLM.
- An OpenShift Container Platform cluster that is managed by the Red Hat Advanced Cluster Management hub cluster
- Access credentials to a local repository where you can mirror the cluster images. See Disconnected installation mirroring for more information about how to create this repository.
  Note: The image for the current version of the cluster that you upgrade must always be available as one of the mirrored images. If an upgrade fails, the cluster reverts back to the version of the cluster at the time that the upgrade was attempted.

1.19.2.2. Prepare your disconnected mirror registry

You must mirror both the image that you want to upgrade to and the current image that you are upgrading from to your local mirror registry. Complete the following steps to mirror the images:

1. Create a script file that contains content that resembles the following example:

```
UPSTREAM_REGISTRY=quay.io
PRODUCT_REPO=openshift-release-dev
RELEASE_NAME=ocp-release
OCP_RELEASE=4.5.2-x86_64
LOCAL_REGISTRY=$(hostname):5000
LOCAL_SECRET_JSON=/path/to/pull/secret
```

```
oc adm -a ${LOCAL_SECRET_JSON} release mirror \
--
from=${UPSTREAM_REGISTRY}/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELE
ASE} \
--to=${LOCAL_REGISTRY}/ocp4 \
--to-release-image=${LOCAL_REGISTRY}/ocp4/release:${OCP_RELEASE}
```

Replace **/path/to/pull/secret** with the path to your OpenShift Container Platform pull secret.

2. Run the script to mirror the images, configure settings, and separate the release images from the release content.

Tip: You can use the output of the last line of this script when you create your **ImageContentSourcePolicy**.

1.19.2.3. Deploy the operator for OpenShift Update Service

To deploy the operator for OpenShift Update Service in your OpenShift Container Platform environment, complete the following steps:

1. On the hub cluster, access the OpenShift Container Platform operator hub.

2. Deploy the operator by selecting **Red Hat OpenShift Update Service Operator**. Update the default values, if necessary. The deployment of the operator creates a new project named **openshift-cincinnati**.

3. Wait for the installation of the operator to finish.
Tip: You can check the status of the installation by entering the **oc get pods** command on your OpenShift Container Platform command line. Verify that the operator is in the **running** state.

1.19.2.4. Build the graph data init container

OpenShift Update Service uses graph data information to determine the available upgrades. In a connected environment, OpenShift Update Service pulls the graph data information for available upgrades directly from the Cincinnati graph data GitHub repository . Because you are configuring a disconnected environment, you must make the graph data available in a local repository by using an **init container**. Complete the following steps to create a graph data **init container**:

1. Clone the *graph data* Git repository by entering the following command:

```
git clone https://github.com/openshift/cincinnati-graph-data
```

2. Create a file that contains the information for your graph data **init**. You can find this sample Dockerfile in the **cincinnati-operator** GitHub repository. The contents of the file is shown in the following sample:

```
FROM registry.access.redhat.com/ubi8/ubi:8.1

RUN curl -L -o cincinnati-graph-data.tar.gz https://github.com/openshift/cincinnati-graph-
data/archive/master.tar.gz

RUN mkdir -p /var/lib/cincinnati/graph-data/

CMD exec /bin/bash -c "tar xvzf cincinnati-graph-data.tar.gz -C /var/lib/
cincinnati/graph-data/ --strip-components=1"
```

In this example:

- The **FROM** value is the external registry where OpenShift Update Service finds the images.
- The **RUN** commands create the directory and package the upgrade files.
- The **CMD** command copies the package file to the local repository and extracts the files for an upgrade.

3. Run the following commands to build the **graph data init container**:

```
podman build -f <path_to_Dockerfile> -t
${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container:latest
podman push ${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-
container:latest --authfile=/path/to/pull_secret.json
```

Replace *path_to_Dockerfile* with the path to the file that you created in the previous step.

Replace *${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container* with the path to your local graph data init container.

Replace */path/to/pull_secret* with the path to your pull secret file.

Note: You can also replace **podman** in the commands with **docker**, if you don't have **podman** installed.

1.19.2.5. Configure certificate for the mirrored registry

If you are using a secure external container registry to store your mirrored OpenShift Container Platform release images, OpenShift Update Service requires access to this registry to build an upgrade graph. Complete the following steps to configure your CA certificate to work with the OpenShift Update Service pod:

1. Find the OpenShift Container Platform external registry API, which is located in **image.config.openshift.io**. This is where the external registry CA certificate is stored. See Configuring additional trust stores for image registry access in the OpenShift Container Platform documentation for more information.
2. Create a ConfigMap in the **openshift-config** namespace.
3. Add your CA certificate under the key **updateservice-registry**. OpenShift Update Service uses this setting to locate your certificate:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: trusted-ca
data:
  updateservice-registry: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

4. Edit the **cluster** resource in the **image.config.openshift.io** API to set the **additionalTrustedCA** field to the name of the ConfigMap that you created.

```
oc patch image.config.openshift.io cluster -p '{"spec":{"additionalTrustedCA":
{"name":"trusted-ca"}}}' --type merge
```

Replace ***trusted-ca*** with the path to your new ConfigMap.

The OpenShift Update Service Operator watches the **image.config.openshift.io** API and the ConfigMap you created in the **openshift-config** namespace for changes, then restart the deployment if the CA cert has changed.

1.19.2.6. Deploy the OpenShift Update Service instance

When you finish deploying the OpenShift Update Service instance on your hub cluster, this instance is located where the images for the cluster upgrades are mirrored and made available to the disconnected managed cluster. Complete the following steps to deploy the instance:

1. If you do not want to use the default namespace of the operator, which is **openshift-cincinnati**, create a namespace for your OpenShift Update Service instance:

   a. In the OpenShift Container Platform hub cluster console navigation menu, select **Administration** > **Namespaces**.

   b. Select **Create Namespace**.

   c. Add the name of your namespace, and any other information for your namespace.

   d. Select **Create** to create the namespace.

2. In the *Installed Operators* section of the OpenShift Container Platform console, select **Red Hat OpenShift Update Service Operator**.

3. Select **Create Instance** in the menu.

4. Paste the contents from your OpenShift Update Service instance. Your YAML instance might resemble the following manifest:

```
apiVersion: cincinnati.openshift.io/v1beta2
kind: Cincinnati
metadata:
  name: openshift-update-service-instance
  namespace: openshift-cincinnati
spec:
  registry: <registry_host_name>:<port>
  replicas: 1
  repository: ${LOCAL_REGISTRY}/ocp4/release
  graphDataImage: '<host_name>:<port>/cincinnati-graph-data-container'
```

   Replace the **spec.registry** value with the path to your local disconnected registry for your images.

   Replace the **spec.graphDataImage** value with the path to your graph data init container. **Tip:** This is the same value that you used when you ran the **podman push** command to push your graph data init container.

5. Select **Create** to create the instance.

6. From the hub cluster CLI, enter the **oc get pods** command to view the status of the instance creation. It might take a while, but the process is complete when the result of the command shows that the instance and the operator are running.

1.19.2.7. Deploy a policy to override the default registry (optional)

Note: The steps in this section only apply if you have mirrored your releases into your mirrored registry.

OpenShift Container Platform has a default image registry value that specifies where it finds the upgrade packages. In a disconnected environment, you can create a policy to replace that value with the path to your local image registry where you mirrored your release images.

For these steps, the policy is named *ImageContentSourcePolicy*. Complete the following steps to create the policy:

1. Log in to the OpenShift Container Platform environment of your hub cluster.
2. In the OpenShift Container Platform navigation, select **Administration** > **Custom Resource Definitions**.
3. Select the *Instances* tab.
4. Select the name of the *ImageContentSourcePolicy* that you created when you set up your disconnected OLM to view the contents.
5. Select the *YAML* tab to view the content in YAML format.
6. Copy the entire contents of the ImageContentSourcePolicy.
7. From the Red Hat Advanced Cluster Management console, select **Governance** > **Create policy**.
8. Set the **YAML** switch to *On* to view the YAML version of the policy.
9. Delete all of the content in the YAML code.
10. Paste the following YAML content into the window to create a custom policy:

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  namespace: default
  annotations:
    policy.open-cluster-management.io/standards: ""
    policy.open-cluster-management.io/categories: ""
    policy.open-cluster-management.io/controls: ""
spec:
  disabled: false
  remediationAction: enforce
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-pod-sample-nginx-pod
          namespace: default
        spec:
```

```
        remediationAction: inform
        severity: low
        object-templates:
          - complianceType: musthave
            objectDefinition:
              apiVersion: operator.openshift.io/v1alpha1
              kind: ImageContentSourcePolicy
              metadata:
                name: <your-local-mirror-name>
              spec:
                repositoryDigestMirrors:
                  - mirrors:
                      - <your-registry>
                    source: registry.redhat.io
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-pod
  namespace: default
placementRef:
  name: placement-policy-pod
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-pod
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-pod
  namespace: default
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      [] # selects all clusters if not specified
```

11. Replace the content inside the **objectDefinition** section of the template with content and add the settings for your **ImageContentSourcePolicy**. Replace **path-to-local-mirror** with the path to your local mirror repository.
Tip: You can find your path to your local mirror by entering the **oc adm release mirror** command.

12. Select the box for **Enforce if supported**.

13. Select **Create** to create the policy.

1.19.2.8. Deploy a policy to deploy a disconnected catalog source

Push the *Catalogsource* policy to the managed cluster to change the default location from a connected location to your disconnected local registry.

1. In the Red Hat Advanced Cluster Management console, select **Infrastructure** > **Clusters**.
2. Find the managed cluster to receive the policy in the list of clusters.
3. Note the value of the **name** label for the managed cluster. The label format is **name=managed-cluster-name**. This value is used when pushing the policy.
4. In the Red Hat Advanced Cluster Management console menu, select **Governance** > **Create policy**.
5. Set the **YAML** switch to *On* to view the YAML version of the policy.
6. Delete all of the content in the **YAML** code.
7. Paste the following **YAML** content into the window to create a custom policy:

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  namespace: default
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-pod-sample-nginx-pod
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: v1
                kind: Pod
                metadata:
                  name: sample-nginx-pod
                  namespace: default
                status:
                  phase: Running
          remediationAction: inform
          severity: low
  remediationAction: enforce
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-pod
  namespace: default
placementRef:
  name: placement-policy-pod
  kind: PlacementRule
```

```
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-pod
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-pod
  namespace: default
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      [] # selects all clusters if not specified
```

8. Add the following content to the policy:

```
apiVersion: config.openshift.io/vi
kind: OperatorHub
metadata:
 name: cluster
spec:
 disableAllDefaultSources: true
```

9. Add the following content:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: <registry_host_name>:<port>/olm/redhat-operators:v1
  displayName: My Operator Catalog
  publisher: grpc
```

   Replace the value of *spec.image* with the path to your local restricted catalog source image.

10. In the Red Hat Advanced Cluster Management console navigation, select **Infrastructure** > **Clusters** to check the status of the managed cluster. When the policy is applied, the cluster status is **ready**.

1.19.2.9. Deploy a policy to change the managed cluster parameter

Push the *ClusterVersion* policy to the managed cluster to change the default location where it retrieves its upgrades.

1. From the managed cluster, confirm that the *ClusterVersion* upstream parameter is currently the default public OpenShift Update Service operand by entering the following command:

```
oc get clusterversion -o yaml
```

The returned content might resemble the following content:

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: ClusterVersion
[..]
  spec:
    channel: stable-4.4
    upstream: https://api.openshift.com/api/upgrades_info/v1/graph
```

2. From the hub cluster, identify the route URL to the OpenShift Update Service operand by entering the following command: **oc get routes**.
   Tip: Note this value for later steps.

3. In the hub cluster Red Hat Advanced Cluster Management console menu, select **Governance** > **Create a policy**.

4. Set the **YAML** switch to *On* to view the YAML version of the policy.

5. Delete all of the content in the **YAML** code.

6. Paste the following **YAML** content into the window to create a custom policy:

```
apiVersion: policy.open-cluster-management.io/v1
kind: Policy
metadata:
  name: policy-pod
  namespace: default
  annotations:
    policy.open-cluster-management.io/standards:
    policy.open-cluster-management.io/categories:
    policy.open-cluster-management.io/controls:
spec:
  disabled: false
  policy-templates:
    - objectDefinition:
        apiVersion: policy.open-cluster-management.io/v1
        kind: ConfigurationPolicy
        metadata:
          name: policy-pod-sample-nginx-pod
        spec:
          object-templates:
            - complianceType: musthave
              objectDefinition:
                apiVersion: v1
                kind: Pod
                metadata:
                  name: sample-nginx-pod
                  namespace: default
                status:
                  phase: Running
          remediationAction: inform
```

```
        severity: low
  remediationAction: enforce
---
apiVersion: policy.open-cluster-management.io/v1
kind: PlacementBinding
metadata:
  name: binding-policy-pod
  namespace: default
placementRef:
  name: placement-policy-pod
  kind: PlacementRule
  apiGroup: apps.open-cluster-management.io
subjects:
- name: policy-pod
  kind: Policy
  apiGroup: policy.open-cluster-management.io
---
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: placement-policy-pod
  namespace: default
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterSelector:
    matchExpressions:
      [] # selects all clusters if not specified
```

7. Add the following content to **policy.spec** in the *policy* section:

```
apiVersion: config.openshift.io/v1
  kind: ClusterVersion
  metadata:
    name: version
  spec:
    channel: stable-4.4
    upstream: https://example-cincinnati-policy-engine-uri/api/upgrades_info/v1/graph
```

Replace the value of *spec.upstream* with the path to your hub cluster OpenShift Update Service operand.

Tip: You can complete the following steps to determine the path to the operand:

a. Run the **oc get get routes -A** command on the hub cluster.

b. Find the route to **cincinnati**. + The path to the operand is the value in the **HOST/PORT** field.

8. In the managed cluster CLI, confirm that the upstream parameter in the **ClusterVersion** is updated with the local hub cluster OpenShift Update Service URL by entering:

```
oc get clusterversion -o yaml
```

Verify that the results resemble the following content:

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: ClusterVersion
[..]
  spec:
    channel: stable-4.4
    upstream: https://<hub-cincinnati-uri>/api/upgrades_info/v1/graph
```

1.19.2.10. Viewing available upgrades

You can view a list of available upgrades for your managed cluster by completing the following steps:

1. Log in to your Red Hat Advanced Cluster Management console.
2. In the navigation menu, select **Infrastructure** > **Clusters**.
3. Select a cluster that is in the *Ready* state.
4. From the **Actions** menu, select **Upgrade cluster**.
5. Verify that the optional upgrade paths are available.
   Note: No available upgrade versions are shown if the current version is not mirrored into the local image repository.

1.19.2.11. Selecting a channel

You can use the Red Hat Advanced Cluster Management console to select a channel for your cluster upgrades on OpenShift Container Platform version 4.6, or later. Those versions must be available on the mirror registry. Complete the steps in Selecting a channel to specify a channel for your upgrades.

1.19.2.12. Upgrading the cluster

After configuring the disconnected registry, Red Hat Advanced Cluster Management and OpenShift Update Service use the disconnected registry to determine if upgrades are available. If no available upgrades are displayed, make sure that you have the release image of the current level of the cluster and at least one later level mirrored in the local repository. If the release image for the current version of the cluster is not available, no upgrades are available.

Complete the following steps to upgrade:

1. In the Red Hat Advanced Cluster Management console, select **Infrastructure** > **Clusters**.
2. Find the cluster that you want to determine if there is an available upgrade.
3. If there is an upgrade available, the **Distribution version** column for the cluster indicates that there is an upgrade available.
4. Select the *Options* menu for the cluster, and select **Upgrade cluster**.
5. Select the target version for the upgrade, and select **Upgrade**.

The managed cluster is updated to the selected version.

If your cluster upgrade fails, the Operator generally retries the upgrade a few times, stops, and reports

the status of the failing component. In some cases, the upgrade process continues to cycle through attempts to complete the process. Rolling your cluster back to a previous version following a failed upgrade is not supported. Contact Red Hat support for assistance if your cluster upgrade fails.

1.20. REMOVING A CLUSTER FROM MANAGEMENT

When you remove an OpenShift Container Platform cluster from management that was created with Red Hat Advanced Cluster Management for Kubernetes, you can either *detach* it or *destroy* it. Detaching a cluster removes it from management, but does not completely delete it. You can import it again if you want to manage it. This is only an option when the cluster is in a *Ready* state.

The following procedures remove a cluster from management in either of the following situations:

- You already deleted the cluster and want to remove the deleted cluster from Red Hat Advanced Cluster Management.
- You want to remove the cluster from management, but have not deleted the cluster.

Important:

- Destroying a cluster removes it from management and deletes the components of the cluster.
- When you detach a managed cluster, the related namespace is automatically deleted. Do not place custom resources in this namespace.
    - Removing a cluster by using the console
    - Removing a cluster by using the command line
    - Removing remaining resources after removing a cluster
    - Defragmenting the etcd database after removing a cluster

1.20.1. Removing a cluster by using the console

From the navigation menu, navigate to **Infrastructure** > **Clusters** and select **Destroy cluster** or **Detach cluster** from the options menu beside the cluster that you want to remove from management.

+ **Tip:** You can detach or destroy multiple clusters by selecting the check boxes of the clusters that you want to detach or destroy and selecting **Detach** or **Destroy**.

Note: If you attempt to detach the hub cluster while it is managed, which is called a **local-cluster**, check to see if the default setting of **disableHubSelfManagement** is **false**. This setting causes the hub cluster to reimport itself and manage itself when it is detached, and it reconciles the **MultiClusterHub** controller. It might take hours for the hub cluster to complete the detachment process and reimport.

To reimport the hub cluster without waiting for the processes to finish, you can enter the following command to restart the **multiclusterhub-operator** pod and reimport faster:

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
```

You can change the value of the hub cluster to not import automatically by changing the **disableHubSelfManagement** value to **true**, as described in Installing while connected online .

1.20.2. Removing a cluster by using the command line

To detach a managed cluster by using the command line of the hub cluster, run the following command:

```
oc delete managedcluster $CLUSTER_NAME
```

To delete the managed cluster after detaching, run the following command:

```
oc delete clusterdeployment <CLUSTER_NAME> -n $CLUSTER_NAME
```

Note: If you attempt to detach the hub cluster, which is named **local-cluster**, be aware that the default setting of **disableHubSelfManagement** is **false**. This setting causes the hub cluster to reimport itself and manage itself when it is detached and it reconciles the **MultiClusterHub** controller. It might take hours for the hub cluster to complete the detachment process and reimport. If you want to reimport the hub cluster without waiting for the processes to finish, you can enter the following command to restart the **multiclusterhub-operator** pod and reimport faster:

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep
multiclusterhub-operator| cut -d' ' -f1`
```

You can change the value of the hub cluster to not import automatically by changing the **disableHubSelfManagement** value to **true**, as described in Installing while connected online .

1.20.3. Removing remaining resources after removing a cluster

If there are remaining resources on the managed cluster that you removed, there are additional steps that are required to ensure that you remove all of the remaining components. Situations when these extra steps are required include the following examples:

- The managed cluster was detached before it was completely created, and components like the **klusterlet** remain on the managed cluster.
- The hub that was managing the cluster was lost or destroyed before detaching the managed cluster, and there is no way to detach the managed cluster from the hub.
- The managed cluster was not in an online state when it was detached.

If one of these situations apply to your attempted detachment of a managed cluster, there are some resources that cannot be removed from managed cluster. Complete the following steps to detach the managed cluster:

1. Make sure you have the **oc** command line interface configured.
2. Make sure you have **KUBECONFIG** configured on your managed cluster.
   If you run **oc get ns | grep open-cluster-management-agent**, you should see two namespaces:

   ```
   open-cluster-management-agent         Active   10m
   open-cluster-management-agent-addon   Active   10m
   ```

3. Run the following command to remove the remaining resources:

   ```
   oc delete namespaces open-cluster-management-agent open-cluster-management-agent-
   addon --wait=false
   oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc delete crds --
   ```

```
wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc patch crds --
type=merge -p '{"metadata":{"finalizers": []}}'
```

4. Run the following command to ensure that both namespaces and all open cluster management **crds** are removed:

```
oc get crds | grep open-cluster-management.io | awk '{print $1}'
oc get ns | grep open-cluster-management-agent
```

1.20.4. Defragmenting the etcd database after removing a cluster

Having many managed clusters can affect the size of the **etcd** database in the hub cluster. In OpenShift Container Platform 4.8, when you delete a managed cluster, the **etcd** database in the hub cluster is not automatically reduced in size. In some scenarios, the **etcd** database can run out of space. An error **etcdserver: mvcc: database space exceeded** is displayed. To correct this error, reduce the size of the **etcd** database by compacting the database history and defragmenting the **etcd** database.

Note: For OpenShift Container Platform version 4.9 and later, the etcd Operator automatically defragments disks and compacts the **etcd** history. No manual intervention is needed. The following procedure is for OpenShift Container Platform version 4.8 and earlier.

Compact the **etcd** history and defragment the **etcd** database in the hub cluster by completing the following procedure.

1.20.4.1. Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

1.20.4.2. Procedure

1. Compact the **etcd** history.
   a. Open a remote shell session to the **etcd** member, for example:

```
$ oc rsh -n openshift-etcd etcd-control-plane-0.example.com etcdctl endpoint status --
cluster -w table
```

   b. Run the following command to compact the **etcd** history:

```
sh-4.4#etcdctl compact $(etcdctl endpoint status --write-out="json" |  egrep -o '"revision":
[0-9]*' | egrep -o '[0-9]*' -m1)
```

   Example output

```
$ compacted revision 158774421
```

2. Defragment the **etcd** database and clear any **NOSPACE** alarms as outlined in Defragmenting **etcd** data.

1.21. CLUSTER BACKUP AND RESTORE OPERATOR

The cluster backup and restore operator provides disaster recovery solutions for when Red Hat Advanced Cluster Management for Kubernetes hub cluster goes down and needs to be recreated. It runs on the hub cluster and depends on the OADP Operator to install Velero, and to create a connection from the hub cluster to the backup storage location where the data is stored. Velero is the component that runs the backup and restore operations. The cluster backup and restore operator solution provides backup and restore support for all Red Hat Advanced Cluster Management hub cluster resources, like managed clusters, applications, policies, and bare metal assets.

It supports backups of any third-party resources that extend the hub cluster installation. With this backup solution, you can define cron-based backup schedules which run at specified time intervals. When the hub cluster goes down, a new hub cluster can be deployed and the backed up data is moved to the new hub cluster.

The cluster backup and restore operator is not installed automatically. Enable the backup component by setting the **cluster-backup** parameter to **true**, in the **MultiClusterHub** resource. The cluster backup operator is installed in the **open-cluster-management-backup** namespace, where Red Hat Advanced Cluster Management is installed. When you install the cluster backup operator, the OADP Operator is also automatically installed.

Notes:

- The OADP Operator 1.0 has disabled building multi-arch builds and only produces **x86_64** builds for the official release. This means that if you are using an architecture other than **x86_64**, the OADP Operator installed by the backup component must be replaced with the correct version. In this case, uninstall the OADP Operator and find the operator matching your architecture, then install it.

- If you have previously installed and used the OADP Operator on the hub cluster, uninstall this version since the backup component works now with OADP installed in the component namespace. Use the same storage location for the **DataProtectionApplication** resource owned by the OADP Operator installed with the backup component; it accesses the same backup data as the previous operator. Velero backup resources are now loaded within the new OADP Operator namespace on this hub cluster.

Velero is installed with the OADP Operator on the Red Hat Advanced Cluster Management hub cluster; Velero is used to backup and restore Red Hat Advanced Cluster Management hub cluster resources.

For a list of supported storage providers for Velero, see S3-Compatible object store providers.

- Prerequisites
- Backup and restore operator architecture
  - Resources that are backed up
    - Extend backup data
    - Resources restored at managed clusters activation time
  - Resource requests and limits customization
  - Protecting data using server-side encryption
  - Schedule a cluster backup
- Restore a backup
  - Prepare the new hub cluster

- Clean the hub cluster before restore
- Resources restored at managed activation time
- Restore passive resources
- Restore passive resources while checking for backups
- Restore imported managed clusters
- Restore activation resources

- Active passive configuration
  - Managed cluster activation data
- Disaster recovery
- Backup validation using a policy

1.21.1. Prerequisites

- Be sure to complete the steps to Create credentials secret for the cloud storage where the backups are saved. The secret resource must be created in the OADP operator namespace, which is **open-cluster-management-backup** namespace.
- Use the created secret when you create a **DataProtectionApplication** resource. Complete the following steps to create an instance of the **DataProtectionApplication** resource:

  1. From the Red Hat OpenShift Container Platform console, select **Operators** > **Installed Operators**.
  2. Click **Create instance** under DataProtectionApplication.
  3. Create the Velero instance by selecting configurations using the {ocp-short) console or by using a YAML file as mentioned in the **DataProtectionApplication** example.
  4. Set the **DataProtectionApplication** namespace to **open-cluster-management-backup**.
  5. Set the specification (**spec:**) values appropriately for the **DataProtectionApplication** resource. Then click **Create**.
     The resource values are mentioned for ease of use. If you intend on using the default backup storage location, set the following value, **default: true** in the **backupStorageLocations** section. Your **DataProtectionApplication** resource might resemble the following YAML file:

     Your **DataProtectionApplication** resource might resemble the following YAML file:

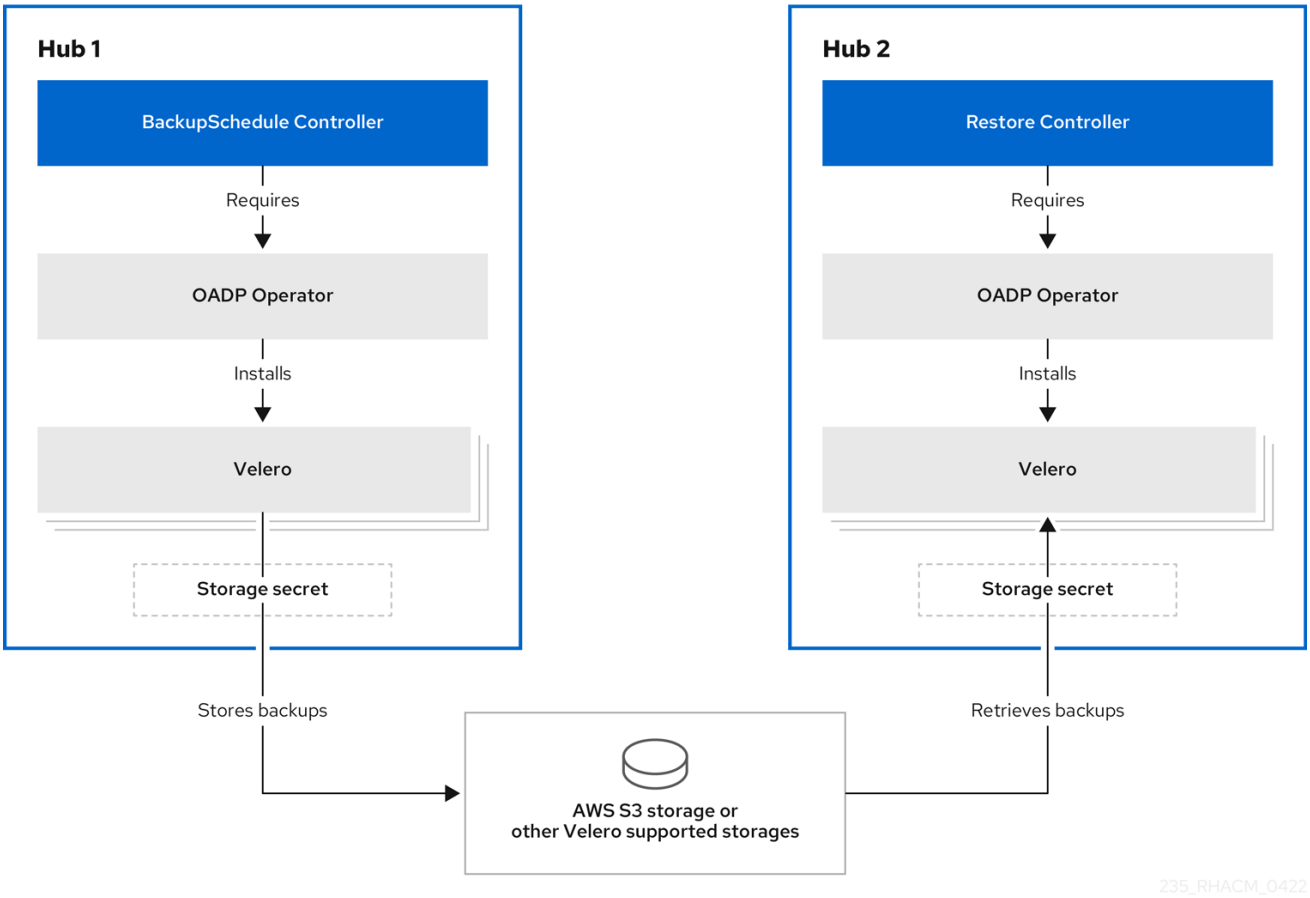
```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: dpa-sample
spec:
  configuration:
    velero:
      defaultPlugins:
```

```
    - openshift
    - aws
  restic:
    enable: true
backupLocations:
  - name: default
    velero:
      provider: aws
      default: true
      objectStorage:
        bucket: my-bucket
        prefix: my-prefix
      config:
        region: us-east-1
        profile: "default"
      credential:
        name: cloud-credentials
        key: cloud
snapshotLocations:
  - name: default
    velero:
      provider: aws
      config:
        region: us-west-2
        profile: "default"
```

See an example to create the **DataProtectionApplication** resource.

1.21.2. Backup and restore operator architecture

The operator defines the **backupSchedule.cluster.open-cluster-management.io** resource, which is used to set up Red Hat Advanced Cluster Management backup schedules, and **restore.cluster.open-cluster-management.io** resource, which is used to process and restore these backups. The operator creates corresponding Velero resources, and defines the options needed to backup remote clusters and any other hub cluster resources that need to be restored. View the following diagram:



1.21.2.1. Resources that are backed up

The cluster backup and restore operator solution provides backup and restore support for all hub cluster resources like managed clusters, applications, policies, and bare metal assets. You can use the solution to back up any third-party resources extending the basic hub cluster installation. With this backup solution, you can define a cron-based backup schedule, which runs at specified time intervals and continuously backs up the latest version of the hub cluster content.

When the hub cluster needs to be replaced or is in a disaster scenario when the hub cluster goes down, a new hub cluster can be deployed and backed up data is moved to the new hub cluster.

View the following ordered list of the cluster backup and restore process for identifying backup data:

- Exclude all resources in the **MultiClusterHub** namespace. This is to avoid backing up installation resources that are linked to the current hub cluster identity and should not be backed up.
- Backup all CRDs with an API version suffixed by **.open-cluster-management.io**. This suffix indicates that all Red Hat Advanced Cluster Management resources are backed up.
- Backup all CRDs from the following API groups: **argoproj.io**, **app.k8s.io**, **core.observatorium.io**, **hive.openshift.io**.
- Exclude all CRDs from the following API groups: **admission.cluster.open-cluster-management.io**, **admission.work.open-cluster-management.io**, **internal.open-cluster-management.io**, **operator.open-cluster-management.io**, **work.open-cluster-management.io**, **search.open-cluster-management.io**, **admission.hive.openshift.io**, **velero.io**.
- Exclude the following CRDs that are a part of the included API groups, but are either not

needed or are being recreated by owner-resources, which are also backed up: **clustermanagementaddon**, **observabilityaddon**, **applicationmanager**, **certpolicycontroller**, **iampolicycontroller**, **policycontroller**, **searchcollector**, **workmanager**, **backupschedule**, **restore**, **clusterclaim.cluster.open-cluster-management.io**.

- Backup secrets and ConfigMaps with one of the following labels: **cluster.open-cluster-management.io/type**, **hive.openshift.io/secret-type**, **cluster.open-cluster-management.io/backup**.

- Use the **cluster.open-cluster-management.io/backup** label for any other resources that you want to be backed up and are not included in the previously mentioned criteria. See the following example:

```
apiVersion: my.group/v1alpha1
kind: MyResource
metadata:
  labels:
    cluster.open-cluster-management.io/backup: ""
```

Note: Secrets used by the **hive.openshift.io.ClusterDeployment** resource need to be backed up, and are automatically annotated with the **cluster.open-cluster-management.io/backup** label only when the cluster is created using the console. If the Hive cluster is deployed using GitOps instead, the **cluster.open-cluster-management.io/backup** label must be manually added to the secrets used by the **ClusterDeployment**.

- Exclude specific resources that you do not want backed up. For example, see the following example to exclude Velero resources from the backup process:

```
apiVersion: my.group/v1alpha1
kind: MyResource
metadata:
  labels:
    velero.io/exclude-from-backup: "true"
```

==== Extend backup data

You can backup third-party resources with cluster backup and restore by adding the **cluster.open-cluster-management.io/backup** label to the resources. The value of the label can be any string, including an empty string. Use a value that can help you identify the component that you are backing up. For example, use the **cluster.open-cluster-management.io/backup: idp** label if the components are provided by an IDP solution.

Note: Use the **cluster-activation** value for the **cluster.open-cluster-management.io/backup** label if you want the resources to be restored when the managed clusters activation resources are restored. Restoring the managed clusters activation resources result in managed clusters being actively managed by the hub cluster, where the restore was started.

1.21.2.1.1. Resources restored at managed clusters activation time

When you add the **cluster.open-cluster-management.io/backup** label to a resource, the resource is automatically backed up in the **acm-resources-generic-schedule** backup. You must set the label value to **cluster-activation** if any of the resources need to be restored, only after the managed clusters are moved to the new hub cluster and when the **veleroManagedClustersBackupName:latest** is used on the restored resource. This ensures the resource is not restored unless the managed cluster activation is called. View the following example:

```
apiVersion: my.group/v1alpha1
kind: MyResource
metadata:
  labels:
    cluster.open-cluster-management.io/backup: cluster-activation
```

Aside from the activation data resources that are identified by using the **cluster.open-cluster-management.io/backup: cluster-activation** label and stored by the **acm-resources-generic-schedule** backup, the cluster backup and restore operator includes a few resources in the activation set, by default. The following resources are backed up by the **acm-managed-clusters-schedule** backup:

- **managedcluster.cluster.open-cluster-management.io**
- **managedcluster.clusterview.open-cluster-management.io**
- **klusterletaddonconfig.agent.open-cluster-management.io**
- **managedclusteraddon.addon.open-cluster-management.io**
- **managedclusterset.cluster.open-cluster-management.io**
- **managedclusterset.clusterview.open-cluster-management.io**
- **managedclustersetbinding.cluster.open-cluster-management.io**
- **clusterpool.hive.openshift.io**
- **clusterclaim.hive.openshift.io**
- **clustercurator.cluster.open-cluster-management.io**

1.21.2.2. Resource requests and limits customization

When Velero is initially installed, Velero pod is set to the default CPU and memory limits as defined in the following sample:

```
resources:
 limits:
   cpu: "1"
   memory: 256Mi
 requests:
   cpu: 500m
   memory: 128Mi
```

The limits from the previous sample work well with some scenarios, but might need to be updated when your cluster backs up a large number of resources. For instance, when back up is run on a hub cluster that manages 2000 clusters, then the Velero pod crashes due to the out-of-memory error (OOM). The following configuration allows for the backup to complete for this scenario:

```
 limits:
   cpu: "2"
   memory: 1Gi
 requests:
   cpu: 500m
   memory: 256Mi
```

To update the limits and requests for the Velero pod resource, you need to update the **DataProtectionApplication** resource and insert the **resourceAllocation** template for the Velero pod. View the following sample:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: velero
  namespace: open-cluster-management-backup
spec:
...
  configuration:
...
    velero:
      podConfig:
        resourceAllocations:
          limits:
            cpu: "2"
            memory: 1Gi
          requests:
            cpu: 500m
            memory: 256Mi
```

Refer to the Velero resource requests and limits customization to find out more about the **DataProtectionApplication** parameters.

1.21.2.3. Protecting data using server-side encryption

Server-side encryption is data encryption for the application or service that receives the data at the storage location. The backup mechanism itself does not encrypt data while in-transit (as it travels to and from backup storage location), or at rest (while it is stored on disks at backup storage location). Instead it relies on the native mechanisms in the object and snapshot systems.

Best practice: Encrypt the data at the destination using the available backup storage server-side encryption. The backup contains resources, such as credentials and configuration files that need to be encrypted when stored outside of the hub cluster.

You can use **serverSideEncryption** and **kmsKeyId** parameters to enable encryption for the backups stored in Amazon S3. For more details, see the Backup Storage Location YAML. The following sample specifies an AWS KMS key ID when setting up the **DataProtectionApplication** resource:

```
spec:
  backupLocations:
    - velero:
        config:
          kmsKeyId: 502b409c-4da1-419f-a16e-eif453b3i49f
          profile: default
          region: us-east-1
```

Refer to Velero supported storage providers to find out about all of the configurable parameters of other storage providers.

1.21.2.4. Schedule a cluster backup

A backup schedule is activated when you create the **backupschedule.cluster.open-cluster-management.io** resource. View the following **backupschedule.cluster.open-cluster-management.io** sample:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: BackupSchedule
metadata:
  name: schedule-acm
spec:
  veleroSchedule: 0 */2 * * *
  veleroTtl: 120h
```

After you create a **backupschedule.cluster.open-cluster-management.io** resource, run the following command to get the status of the scheduled cluster backups:

```
oc get bsch -n <oadp-operator-ns>
```

The **<oadp-operator-ns>** parameter in the previous command is the namespace where the **BackupSchedule** is created, which is the same namespace where the OADP Operator is installed. The **backupschedule.cluster.open-cluster-management.io** resource creates six **schedule.velero.io** resources, which are used to generate backups. Run the following command to view the list of the backups that are scheduled:

```
os get schedules -A | grep acm
```

Resources are separately backed up in the following groups:

- *Credentials backup*, which contains three backup files for Hive, Red Hat Advanced Cluster Management, and user-created credentials.
- *Resources backup*, which contains one backup for the Red Hat Advanced Cluster Management resources and one for generic resources. These resources use the following label, **cluster.open-cluster-management.io/backup**.
- *Managed clusters backup*, which contains only resources that activate the managed cluster connection to the hub cluster, where the backup is restored.

Note: The *resources backup* file contains managed cluster-specific resources, but does not contain the subset of resources that connect managed clusters to the hub cluster. The resources that connect managed clusters are called activation resources and are contained in the managed clusters backup. When you restore backups only for the *credentials* and *resources* backup on a new hub cluster, the new hub cluster shows all managed clusters created with the Hive API in a detached state. However, the managed clusters that are imported on the primary hub cluster using the import operation appear only when the activation data is restored on the passive hub cluster. At this time, the managed clusters are still connected to the original hub cluster that created the backup files.

When the activation data is restored, only managed clusters created using the Hive API are automatically connected with the new hub cluster. All other managed clusters appear in a *Pending* state and must be manually reattached to the new cluster.

1.21.3. Restore a backup

In a usual restore scenario, the hub cluster where the backups are run becomes unavailable, and the backed up data needs to be moved to a new hub cluster. This is done by running the cluster restore operation on the new hub cluster. In this case, the restore operation runs on a different hub cluster than

the one where the backup is created.

There are also cases where you want to restore the data on the same hub cluster where the backup was collected, so the data from a previous snapshot can be recovered. In this case, both restore and backup operations are run on the same hub cluster.

After you create a **restore.cluster.open-cluster-management.io** resource on the hub cluster, you can run the following command to get the status of the restore operation: **oc get restore -n <oadp-operator-ns>**. You should also be able to verify that the backed up resources that are contained by the backup file are created.

Note: The **restore.cluster.open-cluster-management.io** resource is run once. If you want to run the same restore operation again after the restore operation is complete, you have to create a new **restore.cluster.open-cluster-management.io** resource with the same **spec** options.

The restore operation is used to restore all three backup types that are created by the backup operation. However, you can choose to install only a certain type of backup (only managed clusters, only user credentials, or only hub cluster resources).

The restore defines the following three required **spec** properties, where the restore logic is defined for the types of backed up files:

- **veleroManagedClustersBackupName** is used to define the restore option for the managed clusters activation resources.
- **veleroCredentialsBackupName** is used to define the restore option for the user credentials.
- **veleroResourcesBackupName** is used to define the restore option for the hub cluster resources (**Applications**, **Policy**, and other hub cluster resources like managed cluster passive data).
  The valid options for the previously mentioned properties are following values:
  - **latest** - This property restores the last available backup file for this type of backup.
  - **skip** - This property does not attempt to restore this type of backup with the current restore operation.
  - **<backup_name>** - This property restores the specified backup pointing to it by name.

The name of the **restore.velero.io** resources that are created by the **restore.cluster.open-cluster-management.io** is generated using the following template rule, **<restore.cluster.open-cluster-management.io name>-<velero-backup-resource-name>**. View the following descriptions:

- **restore.cluster.open-cluster-management.io name** is the name of the current **restore.cluster.open-cluster-management.io** resource, which initiates the restore.
- **velero-backup-resource-name** is the name of the Velero backup file that is used for restoring the data. For example, the **restore.cluster.open-cluster-management.io** resource named **restore-acm** creates **restore.velero.io** restore resources. View the following examples for the format:
  - **restore-acm-acm-managed-clusters-schedule-20210902205438** is used for restoring managed cluster activation data backups. In this sample, the **backup.velero.io** backup name used to restore the resource is **acm-managed-clusters-schedule-20210902205438**.

- **restore-acm-acm-credentials-schedule-20210902206789** is used for restoring credential backups. In this sample, the **backup.velero.io** backup name used to restore the resource is **acm-managed-clusters-schedule-20210902206789**.

- **restore-acm-acm-resources-schedule-20210902201234** is used for restoring application, policy, and other hub cluster resources like managed cluster passive data backups. In this sample, the **backup.velero.io** backup name used to restore the resource is **acm-managed-clusters-schedule-20210902201234**.

Note: If **skip** is used for a backup type, **restore.velero.io** is not created.

View the following YAML sample of the cluster **Restore** resource. In this sample, all three types of backed up files are being restored, using the latest available backed up files:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
spec:
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

Note: Only managed clusters created by the Hive API are automatically connected with the new hub cluster when the **acm-managed-clusters** backup from the managed clusters backup is restored on another hub cluster. All other managed clusters remain in the **Pending Import** state and must be imported back onto the new hub cluster. For more information, see Restoring imported managed clusters (Technology Preview).

1.21.3.1. Prepare the new hub cluster

Before running the restore operation on a new hub cluster, you need to manually configure the hub cluster and install the same operators as on the initial hub cluster. You must install the Red Hat Advanced Cluster Management operator in the same namespace as the initial hub cluster, create the **DataProtectionApplication** resource, and then connect to the same storage location where the initial hub cluster previously backed up data.

For example, if the initial hub cluster has any other operators installed, such as Ansible Automation Platform, Red Hat OpenShift GitOps, **cert-manager**, you have to install them before running the restore operation. This ensures that the new hub cluster is configured in the same way as the initial hub cluster.

1.21.3.2. Clean the hub cluster before restore

Velero currently skips existing backed up resources on the hub cluster. This limits the scenarios that can be used when you restore hub cluster data on a new hub cluster. If the new hub cluster is used and the restore is applied more than once, the hub cluster is not recommended to use as a passive configuration unless the data is cleaned before restore is ran. The data on the new hub cluster is not reflective of the data available with the restored resources.

When a **restore.cluster.open-cluster-management.io** resource is created, the cluster backup and restore operator runs a set of steps to prepare for restore by cleaning up the hub cluster before the Velero restore begins.

The cleanup option uses the **cleanupBeforeRestore** property to identify the subset of objects to clean up. There are three options you can set for this clean up:

- **None**: No clean up necessary, just begin Velero restore. This is to be used on a brand new hub cluster.

- **CleanupRestored**: Clean up all resources created by a previous Red Hat Advanced Cluster Management restore. It is recommended to use this property because it is less intrusive than the **CleanupAll** property.

- **CleanupAll**: Clean up all resources on the hub cluster, which can be part of an Red Hat Advanced Cluster Management backup, even if the resources are not created as a result of a restore operation. This is to be used when extra content has been created on the hub cluster, which requires clean up. Use this option with caution because this option cleans up resources on the hub cluster created by the user, not by a previous backup. It is strongly recommended to use the **CleanupRestored** option, and to refrain from manually updating hub cluster content when the hub cluster is designated as a passive cluster for a disaster scenario. Use the **CleanupAll** option as a last alternative.

Notes:

- Velero sets the status, **PartiallyFailed**, for a velero restore resource if the restored backup has no resources. This means that a **restore.cluster.open-cluster-management.io** resource can be in **PartiallyFailed** status if any of the created **restore.velero.io** resources do not restore any resources because the corresponding backup is empty.

- The **restore.cluster.open-cluster-management.io** resource is run once, unless you use the **syncRestoreWithNewBackups:true** to keep restoring passive data when new backups are available. For this case, follow the restore passive with sync sample. See Restore passive resources while checking for backups. After the restore operation is complete and you want to run another restore operation on the same hub cluster, you have to create a new **restore.cluster.open-cluster-management.io** resource.

- Although you can create multiple **restore.cluster.open-cluster-management.io** resources, only one can be active at any moment in time.

1.21.3.3. Restore activation resources

Use the **restore-passive-activate** sample when you want the hub cluster to manage the clusters. In this case it is assumed that the other data has been restored already on the hub cluster that using the passive resource.

1.21.3.4. Restore passive resources

Passive data is backup data such as secrets, ConfigMaps, applications, policies, and all the managed cluster custom resources, which do not activate a connection between managed clusters and hub clusters. The backup resources are restored on the hub cluster by the credentials backup and restore resources.

1.21.3.5. Restore passive resources while checking for backups

Use the **restore-passive-sync** sample to restore passive data, while continuing to check if new backups are available and restore them automatically. To automatically restore new backups, you must set the **syncRestoreWithNewBackups** parameter to **true**. You must also only restore the latest passive data.

Set the **VeleroResourcesBackupName** and **VeleroCredentialsBackupName** parameters to **latest**, and the **VeleroManagedClustersBackupName** parameter to **skip**. Immediately after the **VeleroManagedClustersBackupName** is set to **latest**, the managed clusters are activated on the new hub cluster and is now the primary hub cluster.

When the activated managed cluster becomes the primary hub cluster, the restore resource is set to **Finished** and the **syncRestoreWithNewBackups** is ignored, even if set to **true**.

By default, the controler checks for new backups every 30 minutes when the **syncRestoreWithNewBackups** is set to **true**. If new backups are found, it restores the backed up resources. You can change the duration of the check by updating the **restoreSyncInterval** parameter.

For example, the following resource checks for backups every 10 minutes:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm-passive-sync
spec:
  syncRestoreWithNewBackups: true # restore again when new backups are available
  restoreSyncInterval: 10m # check for new backups every 10 minutes
  cleanupBeforeRestore: CleanupRestored
  veleroManagedClustersBackupName: skip
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

1.21.3.6. Restore imported managed clusters

Only managed clusters connected with the primary hub cluster using the Hive API are automatically connected with the new hub cluster, where the activation data is restored. These clusters have been created on the primary hub cluster using the **Create cluster** button in the **Clusters** tab. Managed clusters connected with the initial hub cluster using the **Import cluster** button appear as **Pending Import** when the activation data is restored, and must be imported back on the new hub cluster.

The Hive managed clusters can be connected with the new hub cluster because Hive stores the managed cluster **kubeconfig** in the managed cluster namespace on the hub cluster. This is backed up and restored on the new hub cluster. The import controller then updates the bootstrap **kubeconfig** on the managed cluster using the restored configuration, which is only available for managed clusters created using the Hive API. It is not available for imported clusters.

To reconnect imported clusters on the new hub cluster, manually create the **auto-import-secret** resource after your start the restore operation. See Importing the cluster with the auto import secret for more details.

Create the **auto-import-secret** resource in the managed cluster namespace for each cluster in **Pending Import** state. Use a **kubeconfig** or token with enough permissions for the import component to start the automatic import on the new hub cluster. You must have access for each managed cluster by using a token to connect with the managed cluster. The token must have a **klusterlet** role binding or a role with the same permissions.

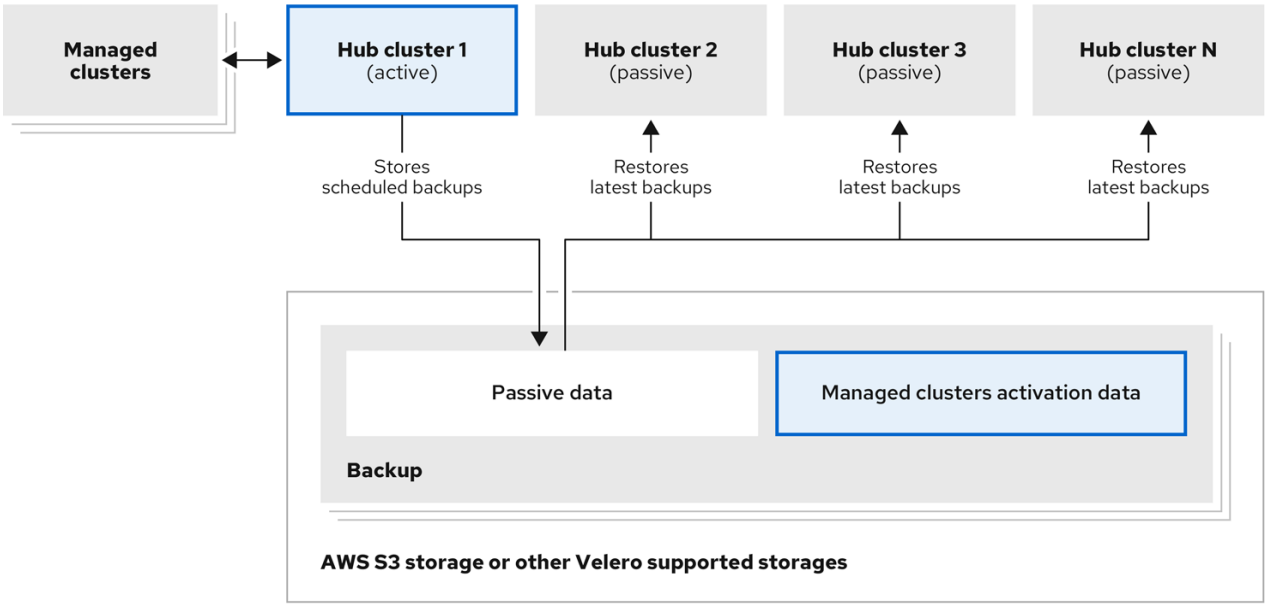
1.21.4. Active passive configuration

In an active passive configuration, there is one active hub cluster and passive hub clusters. An active hub cluster is also considered the primary hub cluster, which manages clusters and backs up resources at defined time intervals, using the **BackupSchedule.cluster.open-cluster-management.io** resource.

Passive hub clusters continuously retrieve the latest backups and restore the passive data. The passive hubs use the **Restore.cluster.open-cluster-management.io** resource to restore passive data from the primary hub cluster when new backup data is available. These hub clusters are on standby to become a primary hub when the primary hub cluster goes down.

Active and passive hub clusters are connected to the same storage location, where the primary hub cluster backs up data for passive hub clusters to access the primary hub cluster backups. For more details on how to setup this automatic restore configuration, see the Restore passive resources while checking for backups section.

In the following diagram, the active hub cluster manages the local clusters and backs up the hub cluster data at regular intervals:



The passive hub cluster restores this data, except for the managed cluster activation data, which moves the managed clusters to the passive hub cluster. The passive hub clusters can restore the passive data continuously, see the Restore passive resources while checking for backups section. Passive hub clusters can restore passive data as a one-time operation, see Restore passive resources section for more details.

1.21.4.1. Managed cluster activation data

Managed cluster activation data or other activation data, is a backup resource. When the activation data is restored on a new hub cluster, managed clusters are then being actively managed by the hub cluster where the restore is run. Activation data resources are stored by the managed clusters backup and by the resource-generic backup, when you use the **cluster.open-cluster-management.io/backup: cluster-activation** label.

1.21.4.2. Resources restored at managed activation time

When you add the **cluster.open-cluster-management.io/backup: cluster-activation** label to a resource, the resource is automatically backed up in the **acm-resources-generic-schedule** backup resource. Resources usually need to be restored when you set the **veleroManagedClustersBackupName:latest** label value in the restore resource. If any of these resources need to be restored when the managed clusters are moved to the new hub cluster, set the **veleroManagedClustersBackupName:latest** label value to **cluster-activation**. This ensures that the resource is not restored unless the managed cluster activation starts.

Your resource might resemble the following example:

```
apiVersion: my.group/v1alpha1
```

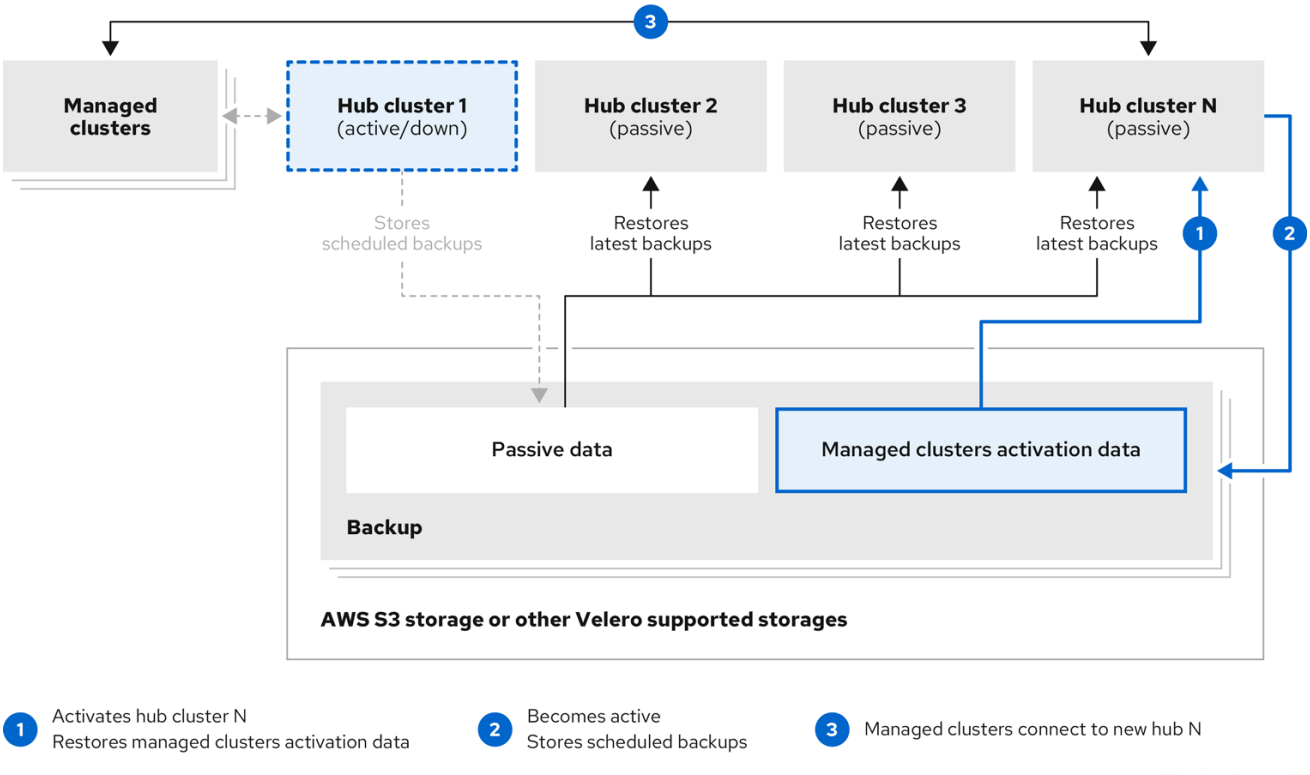
```
kind: MyResource
metadata:
  labels:
    cluster.open-cluster-management.io/backup: cluster-activation
```

There are also default resources in the activation set that are backed up by the **acm-managed-clusters-schedule** resource. View the following default resources that are restored by the **acm-managed-clusters-schedule** resource:

- **managedcluster.cluster.open-cluster-management.io**
- **managedcluster.clusterview.open-cluster-management.io**
- **klusterletaddonconfig.agent.open-cluster-management.io**
- **managedclusteraddon.addon.open-cluster-management.io**
- **clusterpool.hive.openshift.io**
- **clusterclaim.hive.openshift.io**
- **clustercurator.cluster.open-cluster-management.io**
- **clustersync.hiveinternal.openshift.io**
- **baremetalhost.metal3.io**
- **bmceventsubscription.metal3.io**
- **hostfirmwaresettings.metal3.io**

1.21.5. Disaster recovery

When the primary hub cluster goes down, one of the passive hub clusters is chosen by the administrator to take over the managed clusters. In the following image, the administrator decides to use *Hub cluster N* as the new primary hub cluster:



Hub cluster N restores the managed cluster activation data. At this point, the managed clusters connect with *Hub cluster N*. The administrator activates a backup on the new primary hub cluster, *Hub cluster N*, by creating a **BackupSchedule.cluster.open-cluster-management.io** resource, and storing the backups at the same storage location as the initial primary hub cluster.

All other passive hub clusters now restore passive data using the backup data created by the new primary hub cluster. *Hub N* is now the primary hub cluster, managing clusters and backing up data.

1.21.6. Backup validation using a policy

The cluster backup and restore operator Helm chart (**cluster-backup-chart**) installs the **backup-restore-enabled** policy on your hub cluster, which is used to inform you about issues with the backup and restore component. The **backup-restore-enabled** policy includes a set of templates that check for the following constraints:

- Pod validation
  The following templates check the pod status for the backup component and dependencies:
  - **acm-backup-pod-running** template checks if the backup and restore operator pod is running.
  - **oadp-pod-running** template checks if the OADP operator pod is running.
  - **velero-pod-running** template checks if the Velero pod is running.
- Data Protection Application validation
  - **data-protection-application-available** template checks if a **DataProtectioApplicatio.oadp.openshift.io** resource is created. This OADP resource sets up Velero configurations.
- Backup storage validation

  - **backup-storage-location-available** template checks if a **BackupStorageLocation.velero.io** resource is created and if the status value is **Available**. This implies that the connection to the backup storage is valid.

- BackupSchedule collision validation
  - **acm-backup-clusters-collision-report** template verifies that the status is not **BackupCollision**, if a **BackupSchedule.cluster.open-cluster-management.io** exists on the current hub cluster. This verifies that the current hub cluster is not in collision with any other hub cluster when you write backup data to the storage location.
    For a definition of the **BackupCollision** state read the Backup Collisions section .

- BackupSchedule and restore status validation
  - **acm-backup-phase-validation** template checks that the status is not in **Failed**, or **Empty** state, if a **BackupSchedule.cluster.open-cluster-management.io** exists on the current cluster. This ensures that if this cluster is the primary hub cluster and is generating backups, the **BackupSchedule.cluster.open-cluster-management.io** status is healthy.
  - The same template checks that the status is not in a **Failed**, or **Empty** state, if a **Restore.cluster.open-cluster-management.io** exists on the current cluster. This ensures that if this cluster is the secondary hub cluster and is restoring backups, the **Restore.cluster.open-cluster-management.io** status is healthy.

- Backups exist validation
  - **acm-managed-clusters-schedule-backups-available** template checks if **Backup.velero.io** resources are available at the location specified by the **BackupStorageLocation.velero.io**, and if the backups are created by a **BackupSchedule.cluster.open-cluster-management.io** resource. This validates that the backups have been run at least once, using the backup and restore operator.

- Backups for completion
  - An **acm-backup-in-progress-report** template checks if **Backup.velero.io** resources are stuck in the **InProgress** state. This validation is added because with a large number of resources, the velero pod restarts as the backup runs, and the backup stays in progress without proceeding to completion. During a normal backup, the backup resources are in progress at some point when it is run, but are not stuck and run to completion. It is normal to see the **acm-backup-in-progress-report** template report a warning during the time the schedule is running and backups are in progress.

- Backups that actively run as a cron job
  - A **BackupSchedule.cluster.open-cluster-management.io** actively runs and saves new backups at the storage location. This validation is done by the **backup-schedule-cron-enabled** policy template. The template checks that there is a **Backup.velero.io** with **velero.io/schedule-name: acm-validation-policy-schedule** label at the storage location. The **acm-validation-policy-schedule** backups are set to expire after the time is set for the backups cron schedule. If no cron job is running to create backups, the old **acm-validation-policy-schedule** backup is deleted because it expired and a new one is not created. As a result, if no **acm-validation-policy-schedule backups** exist at any moment, it means that there are no active cron jobs generating backups.

    This policy is intended to help notify the hub cluster administrator of any backup issues when the hub cluster is active and produces or restore backups.

Learn how to enable and manage the cluster backup and restore operator, see Manage backup and restore operator.

1.21.7. Manage backup and restore operator

Enable the cluster backup and restore operator to schedule back up and restore for your cluster resources.

Required access: Cluster administrator

- Prerequisites
- Enabling the backup and restore operator
- Using the backup and restore operator
- Viewing restore events

1.21.7.1. Prerequisites

For both active and passive hub clusters

- From your Red Hat OpenShift Container Platform cluster, install the Red Hat Advanced Cluster Management for Kubernetes operator version 2.5.x. The **MultiClusterHub** resource is automatically created when you install Red Hat Advanced Cluster Management, and displays the following status: **Running**.
- The cluster backup and restore operator must be installed manually. Enable the cluster backup and restore operator (**cluster-backup**). Edit the **MultiClusterHub** resource by setting the **cluster-backup** parameter to **true**. This installs the OADP operator in the same namespace with the **cluster-backup** resource.

For passive hub clusters:

- Before you run the restore operation on the passive hub cluster, you must manually configure the hub cluster and install all operators on the active hub cluster, and in the same namespace as the active hub cluster.
- Ensure that the Red Hat Advanced Cluster Management operator is installed in the same namespace as the initial hub cluster. Then create the **DataProtectionApplication** resource and connect to the same storage location where the initial hub cluster backed up data. View the following **DataProtectionApplication** resource sample:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: dpa-sample
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - aws
    restic:
      enable: true
  backupLocations:
```

```
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: my-bucket
          prefix: my-prefix
        config:
          region: us-east-1
          profile: "default"
        credential:
          name: cloud-credentials
          key: cloud
  snapshotLocations:
    - name: default
      velero:
        provider: aws
        config:
          region: us-west-2
          profile: "default"
```

- Before you run the restore operation, verify that other operators, such as Ansible Automation Platform, Red Hat OpenShift Container Platform GitOps, or certificate manager are installed. This ensures that the new hub cluster is configured the same way as the initial hub cluster.

- The passive hub cluster must use the same namespace names as the initial hub cluster when you install the backup and restore operator, and any other operators that are configured on the previous hub cluster.

1.21.7.2. Enabling the backup and restore operator

The cluster backup and restore operator can be enabled when the **MultiClusterHub** resource is created for the first time. The **cluster-backup** parameter is set to **true**. When the operator is enabled, the operator resources are installed.

If the **MultiClusterHub** resource is already created, you can install or uninstall the cluster backup operator by editing the **MultiClusterHub** resource. Set **cluster-backup** to **false**, if you want to uninstall the cluster backup operator.

When the backup and restore operator is enabled, your **MultiClusterHub** resource might resemble the following YAML file:

```
apiVersion: operator.open-cluster-management.io/v1
  kind: MultiClusterHub
  metadata:
    name: multiclusterhub
    namespace: open-cluster-management
  spec:
    availabilityConfig: High
    enableClusterBackup: false
    imagePullSecret: multiclusterhub-operator-pull-secret
    ingress:
      sslCiphers:
        - ECDHE-ECDSA-AES256-GCM-SHA384
        - ECDHE-RSA-AES256-GCM-SHA384
        - ECDHE-ECDSA-AES128-GCM-SHA256
```

```
    - ECDHE-RSA-AES128-GCM-SHA256
overrides:
  components:
    - enabled: true
      name: multiclusterhub-repo
    - enabled: true
      name: search
    - enabled: true
      name: management-ingress
    - enabled: true
      name: console
    - enabled: true
      name: insights
    - enabled: true
      name: grc
    - enabled: true
      name: cluster-lifecycle
    - enabled: true
      name: volsync
    - enabled: true
      name: multicluster-engine
    - enabled: false
      name: cluster-proxy-addon
    - enabled: true <<<<<<<<
      name: cluster-backup
separateCertificateManagement: false
```

1.21.7.3. Using the backup and restore operator

Complete the following steps to schedule and restore backups:

1. Use the backup and restore operator, **backupschedule.cluster.open-cluster-management.io** and **restore.cluster.open-cluster-management.io** resources, to create a **backupschedule.cluster.open-cluster-management.io** resource using the **cluster_v1beta1_backupschedule.yaml** sample file. See the **cluster-backup-operator** samples. Run the following command to create a **backupschedule.cluster.open-cluster-management.io** resource using the **cluster_v1beta1_backupschedule.yaml** sample file:

   ```
   kubectl create -n <oadp-operator-ns> -f
   config/samples/cluster_v1beta1_backupschedule.yaml
   ```

   Your resource might resemble the following file:

   ```
   apiVersion: cluster.open-cluster-management.io/v1beta1
   kind: BackupSchedule
   metadata:
     name: schedule-acm
   spec:
     veleroSchedule: 0 */6 * * * # Create a backup every 6 hours
     veleroTtl: 72h # deletes scheduled backups after 72h; optional, if not specified, the
   maximum default value set by velero is used - 720h
   ```

   View the following descriptions of the **backupschedule.cluster.open-cluster-management.io spec** properties:

- **veleroSchedule** is a required property and defines a cron job for scheduling the backups.
- **veleroTtl** is an optional property and defines the expiration time for a scheduled backup resource. If not specified, the maximum default value set by Velero is used, which is **720h**.

2. Check the status of your **backupschedule.cluster.open-cluster-management.io** resource, which displays the definition for the three **schedule.velero.io** resources. Run the following command:

```
oc get bsch -n <oadp-operator-ns>
```

3. As a reminder, the restore operation is run on a different hub cluster for restore scenarios. To initiate a restore operation, create a **restore.cluster.open-cluster-management.io** resource on the hub cluster where you want to restore backups.
You can use the cluster backup and restore operator, **backupschedule.cluster.open-cluster-management.io** and **restore.cluster.open-cluster-management.io** resources, to create a backup or restore resource. See the **cluster-backup-operator** samples.

4. Run the following command to create a **restore.cluster.open-cluster-management.io** resource using the **cluster_v1beta1_restore.yaml** sample file. Be sure to replace the **oadp-operator-ns** with the namespace name used to install the OADP Operator. The default value for the OADP Operator install namespace is **oadp-operator**:

```
kubectl create -n <oadp-operator-ns> -f config/samples/cluster_v1beta1_restore.yaml
```

Your resource might resemble the following file:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
spec:
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

View the following description of the three required **spec** properties for **restore.cluster.open-cluster-management.io**:

- **veleroManagedClustersBackupName** is used to define the restore option for the managed cluster activiation data.
- **veleroCredentialsBackupName** is used to define the restore option for the user credentials.
- **veleroResourcesBackupName** is used to define the restore option for the hub cluster resources (**Applications**, **Policy**, and other hub resources like managged cluster passive data).
The valid options for the previously mentioned properties are following values:
- **latest** - This property restores the last available backup file for this type of backup.
- **skip** - This property does not attempt to restore this type of backup with the current restore operation.

- **backup_name** - This property restores the specified backup by referencing the name.

5. View the Velero **Restore** resource by running the following command:

```
oc get restore.velero.io -n <oadp-operator-ns>
```

View the following YAML examples to restore different types of backed up files:

- Restore all three types of backed up resources:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
spec:
  veleroManagedClustersBackupSchedule: latest
  veleroCredentialsBackupSchedule: latest
  veleroResourcesBackupSchedule: latest
```

- Restore only managed cluster resources:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
spec:
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: skip
  veleroResourcesBackupName: skip
```

- Restore the resources for managed clusters only, using the **acm-managed-clusters-schedule-20210902205438** backup:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
spec:
  veleroManagedClustersBackupName: acm-managed-clusters-schedule-
20210902205438
  veleroCredentialsBackupName: skip
  veleroResourcesBackupName: skip
```

Notes:

  - The **restore.cluster.open-cluster-management.io** resource is run once. After the restore operation is completed, you can optionally run another restore operation on the same hub cluster. You must create a new **restore.cluster.open-cluster-management.io** resource to run a new restore operation.

  - You can create multiple **restore.cluster.open-cluster-management.io**, however only one can be run at any moment.

1.21.7.4. Viewing restore events

Use the following command to get information about restore events:

```
oc describe -n <oadp-n> <restore-name>
```

Your list of events might resemble the following sample:

```
Spec:
  Cleanup Before Restore:              CleanupRestored
  Restore Sync Interval:               4m
  Sync Restore With New Backups:       true
  Velero Credentials Backup Name:      latest
  Velero Managed Clusters Backup Name:  skip
  Velero Resources Backup Name:        latest
Status:
  Last Message:                    Velero restores have run to completion, restore will continue to sync
with new backups
  Phase:                           Enabled
  Velero Credentials Restore Name:  example-acm-credentials-schedule-20220406171919
  Velero Resources Restore Name:    example-acm-resources-schedule-20220406171920
Events:
  Type    Reason                 Age   From              Message
  ----    ------                 ----  ----              -------
  Normal  Prepare to restore:    76m   Restore controller  Cleaning up resources for backup acm-
credentials-hive-schedule-20220406155817
  Normal  Prepare to restore:    76m   Restore controller  Cleaning up resources for backup acm-
credentials-cluster-schedule-20220406155817
  Normal  Prepare to restore:    76m   Restore controller  Cleaning up resources for backup acm-
credentials-schedule-20220406155817
  Normal  Prepare to restore:    76m   Restore controller  Cleaning up resources for backup acm-
resources-generic-schedule-20220406155817
  Normal  Prepare to restore:    76m   Restore controller  Cleaning up resources for backup acm-
resources-schedule-20220406155817
  Normal  Velero restore created:  74m   Restore controller  example-acm-credentials-schedule-
20220406155817
  Normal  Velero restore created:  74m   Restore controller  example-acm-resources-generic-
schedule-20220406155817
  Normal  Velero restore created:  74m   Restore controller  example-acm-resources-schedule-
20220406155817
  Normal  Velero restore created:  74m   Restore controller  example-acm-credentials-cluster-
schedule-20220406155817
  Normal  Velero restore created:  74m   Restore controller  example-acm-credentials-hive-schedule-
20220406155817
  Normal  Prepare to restore:    64m   Restore controller  Cleaning up resources for backup acm-
resources-schedule-20220406165328
  Normal  Prepare to restore:    62m   Restore controller  Cleaning up resources for backup acm-
credentials-hive-schedule-20220406165328
  Normal  Prepare to restore:    62m   Restore controller  Cleaning up resources for backup acm-
credentials-cluster-schedule-20220406165328
  Normal  Prepare to restore:    62m   Restore controller  Cleaning up resources for backup acm-
credentials-schedule-20220406165328
  Normal  Prepare to restore:    62m   Restore controller  Cleaning up resources for backup acm-
resources-generic-schedule-20220406165328
  Normal  Velero restore created:  61m   Restore controller  example-acm-credentials-cluster-
schedule-20220406165328
  Normal  Velero restore created:  61m   Restore controller  example-acm-credentials-schedule-
```

```
20220406165328
  Normal  Velero restore created:  61m   Restore controller  example-acm-resources-generic-
schedule-20220406165328
  Normal  Velero restore created:  61m   Restore controller  example-acm-resources-schedule-
20220406165328
  Normal  Velero restore created:  61m   Restore controller  example-acm-credentials-hive-schedule-
20220406165328
  Normal  Prepare to restore:      38m   Restore controller  Cleaning up resources for backup acm-
resources-generic-schedule-20220406171920
  Normal  Prepare to restore:      38m   Restore controller  Cleaning up resources for backup acm-
resources-schedule-20220406171920
  Normal  Prepare to restore:      36m   Restore controller  Cleaning up resources for backup acm-
credentials-hive-schedule-20220406171919
  Normal  Prepare to restore:      36m   Restore controller  Cleaning up resources for backup acm-
credentials-cluster-schedule-20220406171919
  Normal  Prepare to restore:      36m   Restore controller  Cleaning up resources for backup acm-
credentials-schedule-20220406171919
  Normal  Velero restore created:  36m   Restore controller  example-acm-credentials-cluster-
schedule-20220406171919
  Normal  Velero restore created:  36m   Restore controller  example-acm-credentials-schedule-
20220406171919
  Normal  Velero restore created:  36m   Restore controller  example-acm-resources-generic-
schedule-20220406171920
  Normal  Velero restore created:  36m   Restore controller  example-acm-resources-schedule-
20220406171920
  Normal  Velero restore created:  36m   Restore controller  example-acm-credentials-hive-schedule-
20220406171919
```

See Restore a backup for a description of the required specification properties and the valid options.