Red Hat Advanced Cluster Management for Kubernetes 2.5

multicluster engine

Read more to learn how to use multicluster engine operator.

Last Updated: 2023-07-13

Red Hat Advanced Cluster Management for Kubernetes 2.5 multicluster engine

Read more to learn how to use multicluster engine operator.

Legal Notice

Abstract

Read more to learn how to use multicluster engine.

Table of Contents

CHAPTER 1. REQUIREMENTS AND RECOMMENDATIONS

Before you install the multicluster engine for Kubernetes operator, review the following system configuration requirements and settings:

- Supported operating systems and platforms
- Network configuration

Important: You must install multicluster engine for Kubernetes on a cluster that does not have Red Hat Advanced Cluster Management for Kubernetes earlier than 2.5 installed. The multicluster engine for Kubernetes cannot co-exist with Red Hat Advanced Cluster Management for Kubernetes on versions earlier than 2.5 because they provide some of the same management components. It is recommended that you install multicluster engine for Kubernetes on a cluster that has never previously installed Red Hat Advanced Cluster Management. If you are using Red Hat Advanced Cluster Management for Kubernetes at version 2.5.0 or later then multicluster engine for Kubernetes will already be installed on the cluster with it.

1.1. SUPPORTED OPERATING SYSTEMS AND PLATFORMS FOR MULTICLUSTER ENGINE FOR KUBERNETES CLUSTERS AND MANAGED CLUSTERS

See the following table for supported operating systems:

Platform	Supported for multicluster engine for Kubernetes cluster	Supported for managed cluster
Red Hat OpenShift Container Platform 3.11.200, and later 3.11.x releases	No	Yes
Red Hat OpenShift Container Platform 4.8.2, and later	Yes	Yes
Red Hat OpenShift Container Platform on Amazon Web Services	Yes	Yes
Red Hat OpenShift Container Platform on Microsoft Azure	Yes	Yes
Red Hat OpenShift Container Platform on Google Cloud Platform	Yes	Yes
Red Hat OpenShift Kubernetes Engine	No	Yes
Google Kubernetes Engine (Google GKE) (Kubernetes 1.17, and later)	No	Yes

Amazon Elastic Kubernetes Service (Amazon EKS) (Kubernetes 1.17.6, and later)	No	Yes
Microsoft Azure Kubernetes Service (Microsoft AKS) (Kubernetes 1.19.6, and later)	No	Yes

1.2. NETWORK CONFIGURATION

Configure your network settings to allow the connections in the following sections:

1.2.1. The multicluster engine for Kubernetes operator cluster networking requirements

For the multicluster engine for Kubernetes cluster networking requirements, see the following table:

Direction	Connection	Port (if specified)
Outbound	API of the cloud provider	
Outbound	(Optional) Kubernetes API server of the provisioned managed cluster	6443
Outbound and inbound	The **WorkManager** service route on the managed cluster	443
Inbound	The Kubernetes API server of the multicluster engine for Kubernetes cluster from the managed cluster	6443
Inbound	Post-commit hook from GitHub to the multicluster engine for Kubernetes cluster. This setting is only required when you use certain application management functions.	6443

1.2.2. Managed cluster networking requirements

For the managed cluster networking requirements, see the following table:

Direction	Connection	Port (if specified)

Direction	Connection	Port (if specified)
Outbound and inbound	Kubernetes API server of the multicluster engine for Kubernetes cluster	6443

1.2.3. Hosted control planes networking requirements (Technology Preview)

When you use hosted control planes, the **HypershiftDeployment** resource must have connectivity to the endpoints listed in the following table:

Direction	Connection	Port (if specified)
Outbound	OpenShift Container Platform control plane and worker nodes	
Outbound	For hosted clusters on Amazon Web Services only: Outbound connection to AWS API and S3 API	
Outbound	For hosted clusters on Microsoft Azure cloud services only: Outbound connection to Azure API	
Outbound	OpenShift Container Platform image repositories that store the ISO images of the coreOS and the image registry for OpenShift Container Platform pods	

CHAPTER 2. GETTING STARTED

2.1. INTRODUCTION

Before you install the multicluster engine for Kubernetes operator, review the system configuration requirements and settings at Requirements and recommendations. With a supported version of OpenShift Container Platform installed and running on your cluster, you can proceed with Installing while connected online.

2.2. CREATE AND MANAGE CLUSTERS

After you have installed, you are ready to create, import, and manage clusters. From your multicluster engine for Kubernetes cluster, you can create other Openshift Container Platform clusters to manage.

1. See Creating a cluster to learn about the types of managed clusters you can create.
2. If you have a cluster that you want to import manually, view Importing a cluster to learn how to import a managed cluster.
3. You can view Detaching a managed cluster when you no longer need to manage a cluster.

2.3. MANIFESTWORK EXAMPLE

You can view the process and example at Deploying workload with ManifestWork.

CHAPTER 3. INSTALLING WHILE CONNECTED ONLINE

The multicluster engine for Kubernetes operator is installed with Operator Lifecycle Manager, which manages the installation, upgrade, and removal of the components that encompass the multicluster engine for Kubernetes engine.

Required access: Cluster administrator

Important:

- You must install multicluster engine for Kubernetes on a cluster that does not have Red Hat Advanced Cluster Management for Kubernetes earlier than 2.5 installed. The multicluster engine for Kubernetes cannot co-exist with Red Hat Advanced Cluster Management for Kubernetes on versions earlier than 2.5 because they provide some of the same management components. It is recommended that you install multicluster engine for Kubernetes on a cluster that has never previously installed Red Hat Advanced Cluster Management. If you are using Red Hat Advanced Cluster Management for Kubernetes at version 2.5.0 or later then multicluster engine for Kubernetes will already be installed on the cluster with it.

- For OpenShift Container Platform Dedicated environment, you must have **cluster-admin** permissions. By default **dedicated-admin** role does not have the required permissions to create namespaces in the OpenShift Container Platform Dedicated environment.

- By default, the engine components are installed on worker nodes of your OpenShift Container Platform cluster without any additional configuration. You can install the engine onto worker nodes by using the OpenShift Container Platform OperatorHub web console interface, or by using the OpenShift Container Platform CLI.

- If you have configured your OpenShift Container Platform cluster with infrastructure nodes, you can install the engine onto those infrastructure nodes by using the OpenShift Container Platform CLI with additional resource parameters. Not all engine components have infrastructure node support, so some worker nodes are still required when installing multicluster engine for Kubernetes on infrastructure nodes. See the *Installing the multicluster engine for Kubernetes engine on infrastructure node* section for those details.

- If you plan to import Kubernetes clusters that were not created by OpenShift Container Platform or multicluster engine for Kubernetes, you will need to configure an image pull secret. For information on how to configure an image pull secret and other advanced configurations, see options in the Advanced configuration section of this documentation.

    - Prerequisites
    - Confirm your OpenShift Container Platform installation
    - Installing from the OperatorHub web console interface
    - Installing from the OpenShift Container Platform CLI
    - Installing the on infrastructure nodes

3.1. PREREQUISITES

Before you install multicluster engine for Kubernetes, see the following requirements:

- Your RedHat OpenShift Container Platform cluster must have access to the multicluster engine for Kubernetes operator in the OperatorHub catalog from the OpenShift Container Platform console.
- You need access to the catalog.redhat.com.
- OpenShift Container Platform version 4.8, or later, must be deployed in your environment, and you must be logged into with the OpenShift Container Platform CLI. See the following install documentation for OpenShift Container Platform:
  - OpenShift Container Platform version 4.10
  - OpenShift Container Platform version 4.9
  - OpenShift Container Platform version 4.8
- Your OpenShift Container Platform command line interface (CLI) must be configured to run **oc** commands. See Getting started with the CLI for information about installing and configuring the OpenShift Container Platform CLI.
- Your OpenShift Container Platform permissions must allow you to create a namespace.
- You must have an Internet connection to access the dependencies for the operator.
- To install in a OpenShift Container Platform Dedicated environment, see the following:
  - You must have the OpenShift Container Platform Dedicated environment configured and running.
  - You must have **cluster-admin** authority to the OpenShift Container Platform Dedicated environment where you are installing the engine.

3.2. CONFIRM YOUR OPENSHIFT CONTAINER PLATFORM INSTALLATION

You must have a supported OpenShift Container Platform version, including the registry and storage services, installed and working. For more information about installing OpenShift Container Platform, see the OpenShift Container Platform documentation.

1. Verify that a multicluster engine for Kubernetes engine operator is not already installed on your OpenShift Container Platform cluster. The multicluster engine for Kubernetes operator allows only one single installation on each OpenShift Container Platform cluster. Continue with the following steps if there is no installation.
2. To ensure that the OpenShift Container Platform cluster is set up correctly, access the OpenShift Container Platform web console with the following command:

   ```
   kubectl -n openshift-console get route
   ```

   See the following example output:

   ```
   openshift-console console console-openshift-console.apps.new-coral.purple-chesterfield.com
   console   https   reencrypt/Redirect     None
   ```

3. Open the URL in your browser and check the result. If the console URL displays **console-openshift-console.router.default.svc.cluster.local**, set the value for

openshift_master_default_subdomain when you install OpenShift Container Platform. See the following example of a URL: **https://console-openshift-console.apps.new-coral.purple-chesterfield.com**.

You can proceed to install multicluster engine for Kubernetes.

3.3. INSTALLING FROM THE OPERATORHUB WEB CONSOLE INTERFACE

Best practice: From the *Administrator* view in your OpenShift Container Platform navigation, install the OperatorHub web console interface that is provided with OpenShift Container Platform.

1. Select **Operators** > **OperatorHub** to access the list of available operators, and select *multicluster engine for Kubernetes* operator.
2. Click **Install**.
3. On the *Operator Installation* page, select the options for your installation:
   - Namespace:
     - The multicluster engine for Kubernetes engine must be installed in its own namespace, or project.
     - By default, the OperatorHub console installation process creates a namespace titled **multicluster-engine**. **Best practice:** Continue to use the **multicluster-engine** namespace if it is available.
     - If there is already a namespace named **multicluster-engine**, choose a different namespace.
   - Channel: The channel that you select corresponds to the release that you are installing. When you select the channel, it installs the identified release, and establishes that the future errata updates within that release are obtained.
   - Approval strategy: The approval strategy identifies the human interaction that is required for applying updates to the channel or release to which you subscribed.
     - Select **Automatic**, which is selected by default, to ensure any updates within that release are automatically applied.
     - Select **Manual** to receive a notification when an update is available. If you have concerns about when the updates are applied, this might be best practice for you.

   Note: To upgrade to the next minor release, you must return to the *OperatorHub* page and select a new channel for the more current release.
4. Select **Install** to apply your changes and create the operator.
5. See the following process to create the *MultiClusterEngine* custom resource.
   a. In the OpenShift Container Platform console navigation, select **Installed Operators** > **multicluster engine for Kubernetes**
   b. Select the **MultiCluster Engine** tab.
   c. Select **Create MultiClusterEngine**.

d. Update the default values in the YAML file. See options in the *MultiClusterEngine advanced configuration* section of the documentation.

- The following example shows the default template that you can copy into the editor:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

6. Select **Create** to initialize the custom resource. It can take up to 10 minutes for the multicluster engine for Kubernetes engine to build and start.
After the *MultiClusterEngine* resource is created, the status for the resource is **Available** on the *MultiCluster Engine* tab.

3.4. INSTALLING FROM THE OPENSHIFT CONTAINER PLATFORM CLI

1. Create a multicluster engine for Kubernetes engine namespace where the operator requirements are contained. Run the following command, where **namespace** is the name for your multicluster engine for Kubernetes engine namespace. The value for **namespace** might be referred to as *Project* in the OpenShift Container Platform environment:

```
oc create namespace <namespace>
```

2. Switch your project namespace to the one that you created. Replace **namespace** with the name of the multicluster engine for Kubernetes engine namespace that you created in step 1.

```
oc project <namespace>
```

3. Create a YAML file to configure an **OperatorGroup** resource. Each namespace can have only one operator group. Replace **default** with the name of your operator group. Replace **namespace** with the name of your project namespace. See the following example:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <default>
spec:
  targetNamespaces:
  - <namespace>
```

4. Run the following command to create the **OperatorGroup** resource. Replace **operator-group** with the name of the operator group YAML file that you created:

```
oc apply -f <path-to-file>/<operator-group>.yaml
```

5. Create a YAML file to configure an OpenShift Container Platform Subscription. Your file should look similar to the following example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
```

```
  name: multicluster-engine
spec:
  sourceNamespace: openshift-marketplace
  source: redhat-operators
  channel: stable-1.0
  installPlanApproval: Automatic
  name: multicluster-engine
```

Note: For installing the multicluster engine for Kubernetes engine on infrastructure nodes, the see Operator Lifecycle Manager Subscription additional configuration section.

6. Run the following command to create the OpenShift Container Platform Subscription. Replace **subscription** with the name of the subscription file that you created:

```
oc apply -f <path-to-file>/<subscription>.yaml
```

7. Create a YAML file to configure the **MultiClusterEngine** custom resource. Your default template should look similar to the following example:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

Note: For installing the multicluster engine for Kubernetes engine on infrastructure nodes, see the MultiClusterEngine custom resource additional configuration section:

8. Run the following command to create the **MultiClusterEngine** custom resource. Replace **custom-resource** with the name of your custom resource file:

```
oc apply -f <path-to-file>/<custom-resource>.yaml
```

If this step fails with the following error, the resources are still being created and applied. Run the command again in a few minutes when the resources are created:

```
error: unable to recognize "./mce.yaml": no matches for kind "MultiClusterEngine" in version
"operator.multicluster-engine.io/v1"
```

9. Run the following command to get the custom resource. It can take up to 10 minutes for the **MultiClusterEngine** custom resource status to display as **Available** in the **status.phase** field after you run the following command:

```
oc get mce -o=jsonpath='{.items[0].status.phase}'
```

If you are reinstalling the multicluster engine for Kubernetes operator and the pods do not start, see Troubleshooting reinstallation failure for steps to work around this problem.

Notes:

- A **ServiceAccount** with a **ClusterRoleBinding** automatically gives cluster administrator privileges to multicluster engine for Kubernetes and to any user credentials with access to the namespace where you install multicluster engine for Kubernetes.

3.5. INSTALLING ON INFRASTRUCTURE NODES

An OpenShift Container Platform cluster can be configured to contain infrastructure nodes for running approved management components. Running components on infrastructure nodes avoids allocating OpenShift Container Platform subscription quota for the nodes that are running those management components.

After adding infrastructure nodes to your OpenShift Container Platform cluster, follow the Installing from the OpenShift Container Platform CLI instructions and add the following configurations to the Operator Lifecycle Manager Subscription and **MultiClusterEngine** custom resource.

3.5.1. Add infrastructure nodes to the OpenShift Container Platform cluster

Follow the procedures that are described in Creating infrastructure machine sets in the OpenShift Container Platform documentation. Infrastructure nodes are configured with a Kubernetes **taint** and **label** to keep non-management workloads from running on them.

To be compatible with the infrastructure node enablement provided by multicluster engine for Kubernetes, ensure your infrastructure nodes have the following **taint** and **label** applied:

```
metadata:
  labels:
    node-role.kubernetes.io/infra: ""
spec:
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
```

3.5.2. Operator Lifecycle Manager Subscription additional configuration

Add the following additional configuration before applying the Operator Lifecycle Manager Subscription:

```
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
      operator: Exists
```

3.5.3. MultiClusterEngine custom resource additional configuration

Add the following additional configuration before applying the **MultiClusterEngine** custom resource:

```
spec:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

CHAPTER 4. INSTALL ON DISCONNECTED NETWORKS

You might need to install the multicluster engine operator on Red Hat OpenShift Container Platform clusters that are not connected to the Internet. The procedure to install on a disconnected engine requires some of the same steps as the connected installation.

Important: You must install multicluster engine for Kubernetes on a cluster that does not have Red Hat Advanced Cluster Management for Kubernetes earlier than 2.5 installed. The multicluster engine for Kubernetes cannot co-exist with Red Hat Advanced Cluster Management for Kubernetes on versions earlier than 2.5 because they provide some of the same management components. It is recommended that you install multicluster engine for Kubernetes on a cluster that has never previously installed Red Hat Advanced Cluster Management. If you are using Red Hat Advanced Cluster Management for Kubernetes at version 2.5.0 or later then multicluster engine for Kubernetes will already be installed on the cluster with it.

You must download copies of the packages to access them during the installation, rather than accessing them directly from the network during the installation.

- Prerequisites
- Confirm your OpenShift Container Platform installation
- Preparing to install the engine on an infrastructure node

4.1. PREREQUISITES

You must meet the following requirements before you install The multicluster engine operator:

- Red Hat OpenShift Container Platform version 4.8 or later must be deployed in your environment, and you must be logged in with the command line interface (CLI).
- You need access to the catalog.redhat.com.
  Note: For managing bare metal clusters, you must have OpenShift Container Platform version 4.8 or later.

  See the OpenShift Container Platform version 4.10, OpenShift Container Platform version 4.8.
- Your Red Hat OpenShift Container Platform CLI must be version 4.8 or later, and configured to run **oc** commands. See Getting started with the CLI for information about installing and configuring the Red Hat OpenShift CLI.
- Your Red Hat OpenShift Container Platform permissions must allow you to create a namespace.
- You must have a workstation with Internet connection to download the dependencies for the operator.

4.2. CONFIRM YOUR OPENSHIFT CONTAINER PLATFORM INSTALLATION

- You must have a supported OpenShift Container Platform version, including the registry and storage services, installed and working in your cluster. For information about OpenShift Container Platform version 4.8, see OpenShift Container Platform documentation.
- When and if you are connected, you can ensure that the OpenShift Container Platform cluster is set up correctly. Access the OpenShift Container Platform web console.

Run the **kubectl -n openshift-console get route** command to access the OpenShift Container Platform web console. See the following example output:

```
openshift-console  console  console-openshift-console.apps.new-coral.purple-chesterfield.com  console  https  reencrypt/Redirect  None
```

The console URL in this example is: **https:// console-openshift-console.apps.new-coral.purple-chesterfield.com**. Open the URL in your browser and check the result.

If the console URL displays **console-openshift-console.router.default.svc.cluster.local**, set the value for **openshift_master_default_subdomain** when you install OpenShift Container Platform.

4.3. INSTALLING IN A DISCONNECTED ENVIRONMENT

Important: You need to download the required images to a mirroring registry to install the operators in a disconnected environment. Without the download, you might receive **ImagePullBackOff** errors during your deployment.

Follow these steps to install the multicluster engine for Kubernetes operator in a disconnected environment:

1. Create a mirror registry. If you do not already have a mirror registry, create one by completing the procedure in the Mirroring images for a disconnected installation topic of the Red Hat OpenShift Container Platform documentation.
   If you already have a mirror registry, you can configure and use your existing one.

2. **Note:** For bare metal only, you need to provide the certificate information for the disconnected registry in your **install-config.yaml** file. To access the image in a protected disconnected registry, you must provide the certificate information so the multicluster engine for Kubernetes operator can access the registry.

   a. Copy the certificate information from the registry.

   b. Open the **install-config.yaml** file in an editor.

   c. Find the entry for **additionalTrustBundle: |**.

   d. Add the certificate information after the **additionalTrustBundle** line. The resulting content should look similar to the following example:

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----
     certificate_content
     -----END CERTIFICATE-----
   sshKey: >-
   ```

3. **Important:** Additional mirrors for disconnected image registries are needed if the following Governance policies are required:

   - Container Security Operator policy: The images are located in the source **registry.redhat.io/quay**.
   - Compliance operator policy: The images are located in the source **registry.redhat.io/compliance**

- Gatekeeper operator policy: The images are located in the source **registry.redhat.io/rhacm2**
  See the following example of mirrors lists for all three operators:

```
- mirrors:
  - <your_registry>/rhacm2
  source: registry.redhat.io/rhacm2
- mirrors:
  - <your_registry>/quay
  source: registry.redhat.io/quay
- mirrors:
  - <your_registry>/compliance
  source: registry.redhat.io/compliance
```

4. Save the **install-config.yaml** file.

5. Create a YAML file that contains the **ImageContentSourcePolicy** with the name **rhacm-policy.yaml**. **Note:** If you modify this on a running cluster, it causes a rolling restart of all nodes.

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mce-repo
spec:
  repositoryDigestMirrors:
  - mirrors:
    - mirror.registry.com:5000/multicluster-engine
    source: registry.redhat.io/multicluster-engine
```

6. Apply the ImageContentSourcePolicy file by entering the following command:

```
oc apply -f mce-policy.yaml
```

7. Enable the disconnected Operator Lifecycle Manager Red Hat Operators and Community Operators.
   the multicluster engine for Kubernetes operator is included in the Operator Lifecycle Manager Red Hat Operator catalog.

8. Configure the disconnected Operator Lifecycle Manager for the Red Hat Operator catalog. Follow the steps in the Using Operator Lifecycle Manager on restricted networks topic of the Red Hat OpenShift Container Platform documentation.

9. Now that you have the image in the disconnected Operator Lifecycle Manager, continue to install the multicluster engine for Kubernetes operator for Kubernetes from the Operator Lifecycle Manager catalog.

See Installing while connected online for the required steps.

CHAPTER 5. CONSOLE OVERVIEW

OpenShift Container Platform console plug-ins are available with OpenShift Container Platform 4.10 web console and can be integrated. To use this feature, the console plug-ins must remain enabled. The multicluster engine operator displays certain console features from **Infrastructure** and **Credentials** navigation items. If you install Red Hat Advanced Cluster Management, you see more console capability.

Note: For OpenShift Container Platform 4.10 with the plug-ins enabled, you can access Red Hat Advanced Cluster Management within the OpenShift Container Platform console from the cluster switcher by selecting **All Clusters** from the drop-down menu.

1. To disable the plug-in, be sure you are in the *Administrator* perspective in the OpenShift Container Platform console.

2. Find **Administration** in the navigation and click **Cluster Settings**, then click *Configuration* tab.

3. From the list of *Configuration resources*, click the **Console** resource with the **operator.openshift.io** API group, which contains cluster-wide configuration for the web console.

4. Click on the *Console plug-ins* tab. The **mce** plug-in is listed. **Note:** If Red Hat Advanced Cluster Management is installed, it is also listed as **acm**.

5. Modify plug-in status from the table. In a few moments, you are prompted to refresh the console.

CHAPTER 6. ADVANCED CONFIGURATION

The multicluster engine for Kubernetes operator is installed using an operator that deploys all of the required components. The multicluster engine for Kubernetes operator can be further configured during or after installation by adding one or more of the following attributes to the **MultiClusterEngine** custom resource:

6.1. CUSTOM IMAGE PULL SECRET

If you plan to import Kubernetes clusters that were not created by OpenShift Container Platform or the multicluster engine for Kubernetes operator, generate a secret that contains your OpenShift Container Platform pull secret information to access the entitled content from the distribution registry.

The secret requirements for OpenShift Container Platform clusters are automatically resolved by OpenShift Container Platform and multicluster engine for Kubernetes, so you do not have to create the secret if you are not importing other types of Kubernetes clusters to be managed.

Important: These secrets are namespace-specific, so make sure that you are in the namespace that you use for your engine.

1. Download your OpenShift Container Platform pull secret file from cloud.redhat.com/openshift/install/pull-secret by selecting **Download pull secret**. Your OpenShift Container Platform pull secret is associated with your Red Hat Customer Portal ID, and is the same across all Kubernetes providers.

2. Run the following command to create your secret:

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

   - Replace **secret** with the name of the secret that you want to create.
   - Replace **namespace** with your project namespace, as the secrets are namespace-specific.
   - Replace **path-to-pull-secret** with the path to your OpenShift Container Platform pull secret that you downloaded.

The following example displays the **spec.imagePullSecret** template to use if you want to use a custom pull secret. Replace **secret** with the name of your pull secret:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  imagePullSecret: <secret>
```

6.2. TARGET NAMESPACE

The operands can be installed in a designated namespace by specifying a location in the **MultiClusterEngine** custom resource. This namespace is created upon application of the **MultiClusterEngine** custom resource.

Important: If no target namespace is specified, the operator will install to the **multicluster-engine** namespace and will set it in the **MultiClusterEngine** custom resource specification.

The following example displays the **spec.targetNamespace** template that you can use to specify a target namespace. Replace **target** with the name of your destination namespace. **Note:** The **target** namespace cannot be the **default** namespace:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  targetNamespace: <target>
```

6.3. AVAILABILITYCONFIG

The Red Hat Advanced Cluster Management hub cluster has two availabilities: **High** and **Basic**. By default, the hub cluster has an availability of **High**, which gives hub cluster components a **replicaCount** of **2**. This provides better support in cases of failover but consumes more resources than the **Basic** availability, which gives components a **replicaCount** of **1**.

The following examples shows the **spec.availabilityConfig** template with **Basic** availability:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  availabilityConfig: "Basic"
```

6.4. NODESELECTOR

You can define a set of node selectors in the **MultiClusterEngine** to install to specific nodes on your cluster. The following example shows **spec.nodeSelector** to assign Red Hat Advanced Cluster Management pods to nodes with the label **node-role.kubernetes.io/infra**:

```
spec:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

6.5. TOLERATIONS

You can define a list of tolerations to allow the **MultiClusterEngine** to tolerate specific taints defined on the cluster. The following example shows a **spec.tolerations** that matches a **node-role.kubernetes.io/infra** taint:

```
spec:
  tolerations:
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule
    operator: Exists
```

The previous infra-node toleration is set on pods by default without specifying any tolerations in the configuration. Customizing tolerations in the configuration will replace this default behavior.

6.6. MANAGEDSERVICEACCOUNT ADD-ON (TECHNOLOGY PREVIEW)

By default, the **Managed-ServiceAccount** add-on is disabled. This component when enabled allows you to create or delete a service account on a managed cluster. To install with this add-on enabled, include the following in the **MultiClusterEngine** specification in **spec.overrides**:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: managedserviceaccount-preview
      enabled: true
```

The **Managed-ServiceAccount** add-on can be enabled after creating **MultiClusterEngine** by editing the resource on the command line and setting the **managedserviceaccount-preview** component to **enabled: true**. Alternatively, you can run the following command and replace <multiclusterengine-name> with the name of your **MultiClusterEngine** resource.

```
oc patch MultiClusterEngine <multiclusterengine-name> --type=json -p='[{"op": "add", "path":
"/spec/overrides/components/-","value":{"name":"managedserviceaccount-preview","enabled":true}}]'
```

See Enabling ManagedServiceAccount add-ons to enable.

6.7. HYPERSHIFT ADD-ON (TECHNOLOGY PREVIEW)

By default, the **Hypershift** add-on is disabled. To install with this add-on enabled, include the following in the **MultiClusterEngine** values in **spec.overrides**:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: hypershift-preview
      enabled: true
```

The **Hypershift** add-on can be enabled after creating **MultiClusterEngine** by editing the resource on the command line, setting the **hypershift-preview** component to **enabled: true**. Alternatively, you can run the following command and replace <multiclusterengine-name> with the name of your **MultiClusterEngine** resource:

```
oc patch MultiClusterEngine <multiclusterengine-name> --type=json -p='[{"op": "add", "path":
"/spec/overrides/components/-","value":{"name":"hypershift-preview","enabled":true}}]'
```

CHAPTER 7. ENABLING MANAGEDSERVICEACCOUNT ADD-ONS (TECHNOLOGY PREVIEW)

When you install the multicluster engine for Kubernetes operator, the **ManagedServiceAccount** add-on is disabled by default. This component when enabled allows you to create or delete a service account on a managed cluster.

Required access: Editor

When a **ManagedServiceAccount** custom resource is created in the **<managed_cluster>** namespace on the hub cluster, a **ServiceAccount** is created on the managed cluster.

A **TokenRequest** is made with the **ServiceAccount** on the managed cluster to the Kubernetes API server on the managed cluster. The token is then stored in a **Secret** in the **<target_managed_cluster>** namespace on the hub cluster.

Note: The token can expire and be rotated. See TokenRequest for more information about token requests.

7.1. PREREQUISITES

- Red Hat OpenShift Container Platform version 4.9 or later must be deployed in your environment, and you must be logged in with the command line interface (CLI).
- You need the multicluster engine for Kubernetes operator installed.

7.2. ENABLING MANAGEDSERVICEACCOUNT

To enable a **Managed-ServiceAccount** add-on for a hub cluster and a managed cluster, complete the following steps:

1. Enable the **ManagedServiceAccount** add-on on hub cluster. See Advanced configuration to learn more.
2. Deploy the **ManagedServiceAccount** add-on and apply it to your target managed cluster. Create the following YAML file and replace **target_managed_cluster** with the name of the managed cluster where you are applying the **Managed-ServiceAccount** add-on:

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: managed-serviceaccount
  namespace: <target_managed_cluster>
spec:
  installNamespace: open-cluster-management-agent-addon
```

3. Run the following command to apply the file:

```
oc apply -f -
```

   You have now enabled the **Managed-ServiceAccount** plugin for your managed cluster. See the following steps to configure a **ManagedServiceAccount**.

4. Create a **ManagedServiceAccount** custom resource with the following YAML source:

```
apiVersion: authentication.open-cluster-management.io/v1alpha1
kind: ManagedServiceAccount
metadata:
  name: <managed_serviceaccount_name>
  namespace: <target_managed_cluster>
spec:
  rotation: {}
```

- Replace **managed_serviceaccount_name** with the name of your **ManagedServiceAccount**.
- Replace **target_managed_cluster** with the name of the managed cluster to which you are applying the **ManagedServiceAccount**.

5. To verify, view the **tokenSecretRef** attribute in the **ManagedServiceAccount** object status to find the secret name and namespace. Run the following command with your account and cluster name:

```
oc get managedserviceaccount <managed_serviceaccount_name> -n
<target_managed_cluster> -o yaml
```

6. View the **Secret** containing the retrieved token that is connected to the created **ServiceAccount** on the managed cluster. Run the following command:

```
oc get secret <managed_serviceaccount_name> -n <target_managed_cluster> -o yaml
```

CHAPTER 8. CREATING A CLUSTER

The multicluster engine for Kubernetes uses internal Hive components to create Red Hat OpenShift Container Platform clusters. See the following information to learn how to create clusters.

- Prerequisites
- Create a cluster with ClusterDeployment
- Create a cluster with cluster pool

8.1. PREREQUISITES

Before creating a cluster, you must clone the clusterImageSets repository and apply it to your hub cluster. See the following steps:

1. Run the following command to clone:

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/release-2.5
```

2. Run the following command to apply it to your hub cluster:

```
find clusterImageSets/fast -type d -exec oc apply -f {} \; 2> /dev/null
```

Select the Red Hat OpenShift Container Platform release images when you create a cluster.

8.2. CREATE A CLUSTER WITH CLUSTERDEPLOYMENT

A **ClusterDeployment** is a Hive custom resource. See the following documentation to learn how to create an individual cluster:

Follow the Using Hive documentation to create the **ClusterDeployment** custom resource.

8.3. CREATE A CLUSTER WITH CLUSTERPOOL

A **ClusterPool** is also a Hive custom resource that is used to create multiple clusters. Create a cluster with the Hive **ClusterPool** API.

Follow the Cluster Pools documentation to provision a cluster.

CHAPTER 9. IMPORTING A CLUSTER

After you install the multicluster engine for Kubernetes operator, you are ready to import a cluster to manage. Using the Red Hat OpenShift Container Platform CLI, you can import a cluster using the **kubeconfig** file of the cluster you are importing. Alternatively, you can run import commands manually on the cluster you are importing. Both procedures are documented.

See the following procedure to import from the CLI:

- Prerequisites
- Prepare for import
- Importing with the auto import secret
- Importing with manual commands
- Detaching a managed cluster

9.1. PREREQUISITES

- You can use Linux (x86_64, s390x, ppc64le) or macOS.
- Ensure you installed the multicluster engine for Kubernetes operator and installed the MultiClusterEngine custom resource on a Kubernetes cluster.
- You need a another separate cluster that you want to manage and Internet connectivity.
- You need the OpenShift Container Platform CLI version 4.8 or later, to run **oc** commands. See Getting started with the OpenShift CLI for information about installing and configuring the Red Hat OpenShift CLI, **oc**.
  Note: Download the installation file for CLI tools from the OpenShift Container Platform console.
- If you are importing a cluster that was not created by OpenShift Container Platform, you need a **multiclusterengine.spec.imagePullSecret** defined. This secret might have been created when multicluster engine for Kubernetes operator was installed. See Custom image pull secret for more information about how to define this secret.

9.2. PREPARE FOR IMPORT

1. Log in to your *engine cluster*. The *engine cluster* is the cluster that contains the the multicluster engine for Kubernetes operator and the custom resource. Run the following command:

   ```
   oc login
   ```

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

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

3. Run the following command to create the namespace:

```
oc label namespace ${CLUSTER_NAME} cluster.open-cluster-
management.io/managedCluster=${CLUSTER_NAME}
```

4. Edit the example **ManagedCluster** with the following sample of YAML:

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

5. **Optional:** In this release, you *cannot* automatically import your hub cluster to become a managed cluster, which is called a **local-cluster**. To manually enable the managed cluster to become a **local-cluster**, add **metadata.labels.local-cluster: "true"**. See the following example YAML and ensure that the name is **local-cluster**. The import will fail or create unexpect results if **local-cluster** is not the name:

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

6. Save the file as **managed-cluster.yaml**.

7. Apply the YAML file with the following command:

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

9.3. IMPORTING WITH THE AUTO IMPORT SECRET

Proceed with the following steps while still logged in to your engine cluster:

1. Retrieve the **kubeconfig** file of the cluster that you are importing 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. Use your **kubeconfig** or your server/token pair to create a YAML file that contains content that is similar to the following template:

```
apiVersion: v1
kind: Secret
metadata:
  name: auto-import-secret
stringData:
  # the following value to specify the retry times when your cluster failed to import
  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 server/token pair, add the following two values:
  server: <cluster_api_url>
  token: <Token to access the cluster>
type: Opaque
```

3. Save the file as auto-import-secret.yaml

4. Generate the import secret in the **${CLUSTER_NAME}** namespace using the **kubeconfig** file of the cluster you are importing. Run the following command with your path to your **kubeconfig** and **CLUSTER_NAME**:

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

   Note: The auto import secret is used one time and deleted when the import process completes.

5. Validate the **JOINED** and **AVAILABLE** status for your imported cluster. Run the following command from the multicluster engine for Kubernetes cluster:

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

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

6. Log in to the cluster that you are importing. Run the following command:

```
oc login
```

7. Validate the pod status on the cluster that you are importing. Run the following command:

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

8. Add-ons will be installed after the cluster that you are importing is **AVAILABLE**. Validate the pod status of add-ons on the cluster. Run the following command:

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

Your cluster is now imported.

9.4. IMPORTING WITH MANUAL COMMAND

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** that was generated by the import controller on your engine cluster. Run 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** that was generated by the import controller on your engine cluster. Run 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 separate cluster that you are importing:

3. Log in to the cluster that you are importing.

```
oc login
```

4. Apply the **klusterlet-crd.yaml** that you generated in the previous step. Run the following command:

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

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

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

6. Validate the pod status on the cluster you are importing. Run the following command:

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

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

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

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

8. Validate the pod status of add-ons on the cluster you are importing. Run the following command:

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

Your cluster is now imported and you can manage that cluster from the engine cluster.

9.5. DETACHING A MANAGED CLUSTER

A managed cluster is a cluster that was successfully imported. To detach a managed cluster from the engine cluster, run the following command:

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

Your cluster is now detached.

CHAPTER 10. DEPLOYING WORKLOAD WITH MANIFESTWORK

You can deploy workloads onto your managed clusters from your multicluster engine for Kubernetes cluster. For example. See the following sample with **ManifestWork** to create a basic deployment on your managed cluster from your multicluster engine for Kubernetes cluster:

1. Log in to your multicluster engine for Kubernetes cluster:

```
oc login
```

2. Create a YAML file to configure the **ManifestWork** resource as it is in the following example. Replace **CLUSTER_NAME** with the name of the managed cluster that you imported from the Importing a cluster documentation. The example YAML deploys to your managed cluster **default** namespace when you apply the file:

```
apiVersion: work.open-cluster-management.io/v1
kind: ManifestWork
metadata:
  name: hello-work
  namespace: ${CLUSTER_NAME}
  labels:
    app: hello
spec:
  workload:
    manifests:
    - apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: hello
        namespace: default
      spec:
        selector:
          matchLabels:
            app: hello
        template:
          metadata:
            labels:
              app: hello
          spec:
            containers:
              - name: hello
                image: quay.io/asmacdo/busybox
                command: ['/bin/sh', '-c', 'echo "Hello, Kubernetes!" && sleep 300']
    - apiVersion: v1
      kind: Service
      metadata:
        labels:
          app: hello
        name: hello
        namespace: default
      spec:
        ports:
        - port: 8000
          protocol: TCP
```

```
    targetPort: 8000
  selector:
    app: hello
```

3. Apply the YAML file. Run the following command:

```
oc apply -f manifestwork.yaml
```

4. Run the following command to check the status of the **ManifestWork** from your multicluster engine for Kubernetes cluster:

```
oc get manifestwork -n ${CLUSTER_NAME} hello-work -o yaml
```

5. Log in to your managed cluster to view the results. See the following command:

```
oc login
```

6. View the deployment that you created with your multicluster engine for Kubernetes cluster:

```
$ oc get deploy -n default
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
hello   1/1     1            1           37s
```

You can also view the created pod with the following command:

```
$ oc get pod
NAME                     READY   STATUS    RESTARTS   AGE
hello-65f58985ff-4rm57   1/1     Running   0          42s
```

If you view the logs for the created pod, you see a message similar to the following:

```
$ oc logs hello-65f58985ff-4rm57
Hello, Kubernetes!
```

CHAPTER 11. APIS

You can access APIs for the multicluster engine for Kubernetes operator for cluster lifecycle management. **User required access:** You can only perform actions that your role is assigned. For more information, review the API documentation for each of the following resources:

- Clusters API
- ClusterSets API (v1beta1)
- Clusterview API
- ClusterSetBindings API (v1beta1)
- MultiClusterEngine API
- Placements API (v1alpha1)
- PlacementDecisions API (v1alpha1)
- Managed service account (Technology Preview)

11.1. CLUSTERS API

11.1.1. Overview

This documentation is for the cluster resource for multicluster engine for Kubernetes. Cluster resource has four possible requests: create, query, delete and update.

11.1.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.1.1.2. Tags

- cluster.open-cluster-management.io : Create and manage clusters

11.1.2. Paths

11.1.2.1. Query all clusters

```
GET /cluster.open-cluster-management.io/v1/managedclusters
```

11.1.2.1.1. Description

Query your clusters for more details.

11.1.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

11.1.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.1.2.1.4. Consumes

- **cluster/yaml**

11.1.2.1.5. Tags

- cluster.open-cluster-management.io

11.1.2.2. Create a cluster

```
POST /cluster.open-cluster-management.io/v1/managedclusters
```

11.1.2.2.1. Description

Create a cluster

11.1.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the cluster to be created.	Cluster

11.1.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.1.2.2.4. Consumes

- **cluster/yaml**

11.1.2.2.5. Tags

- cluster.open-cluster-management.io

11.1.2.2.6. Example HTTP request

11.1.2.2.6.1. Request body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1",
  "kind" : "ManagedCluster",
  "metadata" : {
    "labels" : {
      "vendor" : "OpenShift"
    },
    "name" : "cluster1"
  },
  "spec": {
    "hubAcceptsClient": true,
    "managedClusterClientConfigs": [
      {
        "caBundle": "test",
        "url": "https://test.com"
      }
    ]
  },
  "status" : { }
}
```

11.1.2.3. Query a single cluster

```
GET /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}
```

11.1.2.3.1. Description

Query a single cluster for more details.

11.1.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**cluster_name** *required*	Name of the cluster that you want to query.	string

11.1.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.1.2.3.4. Tags

- cluster.open-cluster-management.io

11.1.2.4. Delete a cluster

```
DELETE /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}
```

11.1.2.4.1. Description

Delete a single cluster

11.1.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**cluster_name** *required*	Name of the cluster that you want to delete.	string

11.1.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.1.2.4.4. Tags

- cluster.open-cluster-management.io

11.1.3. Definitions

11.1.3.1. Cluster

Name	Schema
apiVersion *required*	string
kind *required*	string
metadata *required*	object
spec *required*	spec

spec

Name	Schema
hubAcceptsClient *required*	bool
managedClusterClientConfigs *optional*	< managedClusterClientConfigs > array
leaseDurationSeconds *optional*	integer (int32)

managedClusterClientConfigs

Name	Description	Schema
URL *required*		string
CABundle *optional*	Pattern : **"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"**	string (byte)

11.2. CLUSTERSETS API (V1ALPHA1)

11.2.1. Overview

This documentation is for the Clusterset resource for multicluster engine for Kubernetes. Clusterset resource has four possible requests: create, query, delete and update.

11.2.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.2.1.2. Tags

- cluster.open-cluster-management.io : Create and manage Clustersets

11.2.2. Paths

11.2.2.1. Query all clustersets

```
GET /cluster.open-cluster-management.io/v1beta1/managedclustersets
```

11.2.2.1.1. Description

Query your Clustersets for more details.

11.2.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

11.2.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.2.2.1.4. Consumes

- **clusterset/yaml**

11.2.2.1.5. Tags

- cluster.open-cluster-management.io

11.2.2.2. Create a clusterset

```
POST /cluster.open-cluster-management.io/v1beta1/managedclustersets
```

11.2.2.2.1. Description

Create a Clusterset.

11.2.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the clusterset to be created.	Clusterset

11.2.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.2.2.2.4. Consumes

- **clusterset/yaml**

11.2.2.2.5. Tags

- cluster.open-cluster-management.io

11.2.2.2.6. Example HTTP request

11.2.2.2.6.1. Request body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta1",
  "kind" : "ManagedClusterSet",
  "metadata" : {
    "name" : "clusterset1"
  },
  "spec": { },
  "status" : { }
}
```

11.2.2.3. Query a single clusterset

```
GET /cluster.open-cluster-management.io/v1beta1/managedclustersets/{clusterset_name}
```

11.2.2.3.1. Description

Query a single clusterset for more details.

11.2.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterset_name** *required*	Name of the clusterset that you want to query.	string

11.2.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.2.2.3.4. Tags

- cluster.open-cluster-management.io

11.2.2.4. Delete a clusterset

```
DELETE /cluster.open-cluster-management.io/v1beta1/managedclustersets/{clusterset_name}
```

11.2.2.4.1. Description

Delete a single clusterset.

11.2.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**clusterset_name** *required*	Name of the clusterset that you want to delete.	string

11.2.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.2.2.4.4. Tags

- cluster.open-cluster-management.io

11.2.3. Definitions

11.2.3.1. Clusterset

Name	Schema
apiVersion *required*	string
kind *required*	string
metadata *required*	object

11.3. CLUSTERVIEW API (V1ALPHA1)

11.3.1. Overview

This documentation is for the **clusterview** resource for multicluster engine for Kubernetes. The **clusterview** resource provides a CLI command that enables you to view a list of the managed clusters and managed cluster sets that that you can access. The three possible requests are: list, get, and watch.

11.3.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.3.1.2. Tags

- clusterview.open-cluster-management.io : View a list of managed clusters that your ID can access.

11.3.2. Paths

11.3.2.1. Get managed clusters

```
GET /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.1.1. Description

View a list of the managed clusters that you can access.

11.3.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

11.3.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.3.2.1.4. Consumes

- **managedcluster/yaml**

11.3.2.1.5. Tags

- clusterview.open-cluster-management.io

11.3.2.2. List managed clusters

```
LIST /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.2.1. Description

View a list of the managed clusters that you can access.

11.3.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *optional*	Name of the user ID for which you want to list the managed clusters.	string

11.3.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.3.2.2.4. Consumes

- **managedcluster/yaml**

11.3.2.2.5. Tags

- clusterview.open-cluster-management.io

11.3.2.2.6. Example HTTP request

11.3.2.2.6.1. Request body

```
{
  "apiVersion" : "clusterview.open-cluster-management.io/v1alpha1",
  "kind" : "ClusterView",
  "metadata" : {
    "name" : "<user_ID>"
  },
  "spec": { },
  "status" : { }
}
```

11.3.2.3. Watch the managed cluster sets

```
WATCH /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.3.1. Description

Watch the managed clusters that you can access.

11.3.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

11.3.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.3.2.4. List the managed cluster sets.

```
GET /managedclustersets.clusterview.open-cluster-management.io
```

11.3.2.4.1. Description

List the managed clusters that you can access.

11.3.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

11.3.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.3.2.5. List the managed cluster sets.

```
LIST /managedclustersets.clusterview.open-cluster-management.io
```

11.3.2.5.1. Description

List the managed clusters that you can access.

11.3.2.5.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

11.3.2.5.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.3.2.6. Watch the managed cluster sets.

```
WATCH /managedclustersets.clusterview.open-cluster-management.io
```

11.3.2.6.1. Description

Watch the managed clusters that you can access.

11.3.2.6.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

11.3.2.6.3. Responses

HTTP Code	Description	Schema
200	Success	No Content

HTTP Code	Description	Schema
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.4. CLUSTERSETBINDINGS API (V1ALPHA1)

11.4.1. Overview

This documentation is for the clustersetbinding resource for multicluster engine for Kubernetes. Clustersetbinding resource has four possible requests: create, query, delete and update.

11.4.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.4.1.2. Tags

- cluster.open-cluster-management.io : Create and manage clustersetbindings

11.4.2. Paths

11.4.2.1. Query all clustersetbindings

```
GET /cluster.open-cluster-
management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings
```

11.4.2.1.1. Description

Query your clustersetbindings for more details.

11.4.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string

11.4.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.4.2.1.4. Consumes

- **clustersetbinding/yaml**

11.4.2.1.5. Tags

- cluster.open-cluster-management.io

11.4.2.2. Create a clustersetbinding

```
POST /cluster.open-cluster-
management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings
```

11.4.2.2.1. Description

Create a clustersetbinding.

11.4.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string

Type	Name	Description	Schema
Body	**body** *required*	Parameters describing the clustersetbinding to be created.	Clustersetbinding

11.4.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.4.2.2.4. Consumes

- **clustersetbinding/yaml**

11.4.2.2.5. Tags

- cluster.open-cluster-management.io

11.4.2.2.6. Example HTTP request

11.4.2.2.6.1. Request body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1",
  "kind" : "ManagedClusterSetBinding",
  "metadata" : {
    "name" : "clusterset1",
    "namespace" : "ns1"
  },
 "spec": {
    "clusterSet": "clusterset1"
  },
  "status" : { }
}
```

11.4.2.3. Query a single clustersetbinding

> GET /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings/{clustersetbinding_name}

11.4.2.3.1. Description

Query a single clustersetbinding for more details.

11.4.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string
Path	**clustersetbinding_name** *required*	Name of the clustersetbinding that you want to query.	string

11.4.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.4.2.3.4. Tags

- cluster.open-cluster-management.io

11.4.2.4. Delete a clustersetbinding

> DELETE /cluster.open-cluster-management.io/v1beta1/managedclustersetbindings/{clustersetbinding_name}

11.4.2.4.1. Description

Delete a single clustersetbinding.

11.4.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string
Path	**clustersetbinding_name** *required*	Name of the clustersetbinding that you want to delete.	string

11.4.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.4.2.4.4. Tags

- cluster.open-cluster-management.io

11.4.3. Definitions

11.4.3.1. Clustersetbinding

Name	Schema
apiVersion *required*	string
kind *required*	string

Name	Schema
metadata *required*	object
spec *required*	spec

spec

Name	Schema
clusterSet *required*	string

11.5. API

11.5.1. Overview

This documentation is for the MultiClusterEngine resource for multicluster engine for Kubernetes. The **MultiClusterEngine** resource has four possible requests: create, query, delete, and update.

11.5.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.5.1.2. Tags

- multiclusterengines.multicluster.openshift.io : Create and manage MultiClusterEngines

11.5.2. Paths

11.5.2.1. Create a MultiClusterEngine

```
POST /apis/multicluster.openshift.io/v1alpha1/multiclusterengines
```

11.5.2.1.1. Description

Create a MultiClusterEngine.

11.5.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the MultiClusterEngine to be created.	MultiClusterEngine

11.5.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.5.2.1.4. Consumes

- **MultiClusterEngines/yaml**

11.5.2.1.5. Tags

- multiclusterengines.multicluster.openshift.io

11.5.2.1.5.1. Request body

```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "multiclusterengines.multicluster.openshift.io"
  },
  "spec": {
    "group": "multicluster.openshift.io",
    "names": {
      "kind": "MultiClusterEngine",
      "listKind": "MultiClusterEngineList",
      "plural": "multiclusterengines",
      "shortNames": [
```

```
        "mce"
      ],
      "singular": "multiclusterengine"
    },
    "scope": "Cluster",
    "versions": [
      {
        "additionalPrinterColumns": [
          {
            "description": "The overall state of the MultiClusterEngine",
            "jsonPath": ".status.phase",
            "name": "Status",
            "type": "string"
          },
          {
            "jsonPath": ".metadata.creationTimestamp",
            "name": "Age",
            "type": "date"
          }
        ],
        "name": "v1alpha1",
        "schema": {
          "openAPIV3Schema": {
            "description": "MultiClusterEngine is the Schema for the multiclusterengines\nAPI",
            "properties": {
              "apiVersion": {
                "description": "APIVersion defines the versioned schema of this representation\nof an
object. Servers should convert recognized schemas to the latest\ninternal value, and may reject
unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#resources",
                "type": "string"
              },
              "kind": {
                "description": "Kind is a string value representing the REST resource this\nobject
represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be
updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#types-kinds",
                "type": "string"
              },
              "metadata": {
                "type": "object"
              },
              "spec": {
                "description": "MultiClusterEngineSpec defines the desired state of MultiClusterEngine",
                "properties": {
                  "imagePullSecret": {
                    "description": "Override pull secret for accessing MultiClusterEngine\noperand and
endpoint images",
                    "type": "string"
                  },
                  "nodeSelector": {
                    "additionalProperties": {
                      "type": "string"
                    },
                    "description": "Set the nodeselectors",
                    "type": "object"
```

```
            },
            "targetNamespace": {
              "description": "Location where MCE resources will be placed",
              "type": "string"
            },
            "tolerations": {
              "description": "Tolerations causes all components to tolerate any taints.",
              "items": {
                "description": "The pod this Toleration is attached to tolerates any\ntaint that matches
the triple <key,value,effect> using the matching\noperator <operator>.",
                "properties": {
                 "effect": {
                   "description": "Effect indicates the taint effect to match. Empty\nmeans match all taint
effects. When specified, allowed values\nare NoSchedule, PreferNoSchedule and NoExecute.",
                   "type": "string"
                 },
                 "key": {
                   "description": "Key is the taint key that the toleration applies\nto. Empty means match
all taint keys. If the key is empty,\noperator must be Exists; this combination means to match
all\nvalues and all keys.",
                   "type": "string"
                 },
                 "operator": {
                   "description": "Operator represents a key's relationship to the\nvalue. Valid operators
are Exists and Equal. Defaults to Equal.\nExists is equivalent to wildcard for value, so that a pod\ncan
tolerate all taints of a particular category.",
                   "type": "string"
                 },
                 "tolerationSeconds": {
                   "description": "TolerationSeconds represents the period of time\nthe toleration (which
must be of effect NoExecute, otherwise\nthis field is ignored) tolerates the taint. By default, it\nis not
set, which means tolerate the taint forever (do not\nevict). Zero and negative values will be treated as
0 (evict\nimmediately) by the system.",
                   "format": "int64",
                   "type": "integer"
                 },
                 "value": {
                   "description": "Value is the taint value the toleration matches\nto. If the operator is
Exists, the value should be empty,\notherwise just a regular string.",
                   "type": "string"
                 }
               },
               "type": "object"
             },
             "type": "array"
           }
         },
         "type": "object"
       },
       "status": {
         "description": "MultiClusterEngineStatus defines the observed state of MultiClusterEngine",
         "properties": {
           "components": {
             "items": {
               "description": "ComponentCondition contains condition information for\ntracked
components",
```

```
                    "properties": {
                     "kind": {
                      "description": "The resource kind this condition represents",
                      "type": "string"
                     },
                     "lastTransitionTime": {
                      "description": "LastTransitionTime is the last time the condition\nchanged from one
status to another.",
                      "format": "date-time",
                      "type": "string"
                     },
                     "message": {
                      "description": "Message is a human-readable message indicating\ndetails about the
last status change.",
                      "type": "string"
                     },
                     "name": {
                      "description": "The component name",
                      "type": "string"
                     },
                     "reason": {
                      "description": "Reason is a (brief) reason for the condition's\nlast status change.",
                      "type": "string"
                     },
                     "status": {
                      "description": "Status is the status of the condition. One of True,\nFalse, Unknown.",
                      "type": "string"
                     },
                     "type": {
                      "description": "Type is the type of the cluster condition.",
                      "type": "string"
                     }
                    },
                    "type": "object"
                   },
                   "type": "array"
                  },
                  "conditions": {
                   "items": {
                    "properties": {
                     "lastTransitionTime": {
                      "description": "LastTransitionTime is the last time the condition\nchanged from one
status to another.",
                      "format": "date-time",
                      "type": "string"
                     },
                     "lastUpdateTime": {
                      "description": "The last time this condition was updated.",
                      "format": "date-time",
                      "type": "string"
                     },
                     "message": {
                      "description": "Message is a human-readable message indicating\ndetails about the
last status change.",
                      "type": "string"
                     },
```

```
                    "reason": {
                      "description": "Reason is a (brief) reason for the condition's\nlast status change.",
                      "type": "string"
                    },
                    "status": {
                      "description": "Status is the status of the condition. One of True,\nFalse, Unknown.",
                      "type": "string"
                    },
                    "type": {
                      "description": "Type is the type of the cluster condition.",
                      "type": "string"
                    }
                  },
                  "type": "object"
                },
                "type": "array"
              },
              "phase": {
                "description": "Latest observed overall state",
                "type": "string"
              }
            },
            "type": "object"
          }
        },
        "type": "object"
      }
    },
    "served": true,
    "storage": true,
    "subresources": {
      "status": {}
    }
  }
 ]
},
"status": {
 "acceptedNames": {
  "kind": "",
  "plural": ""
 },
 "conditions": [],
 "storedVersions": []
}
}
```


11.5.2.2. Query all MultiClusterEngines

```
GET /apis/multicluster.openshift.io/v1alpha1/multiclusterengines
```

11.5.2.2.1. Description

Query your multicluster engine for more details.

11.5.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

11.5.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.5.2.2.4. Consumes

- **operator/yaml**

11.5.2.2.5. Tags

- multiclusterengines.multicluster.openshift.io

11.5.2.3. Delete a MultiClusterEngine operator

```
DELETE /apis/multicluster.openshift.io/v1alpha1/multiclusterengines/{name}
```

11.5.2.3.1. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**name** *required*	Name of the multiclusterengine that you want to delete.	string

11.5.2.3.2. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.5.2.3.3. Tags

- multiclusterengines.multicluster.openshift.io

11.5.3. Definitions

11.5.3.1. MultiClusterEngine

Name	Description	Schema
apiVersion *required*	The versioned schema of the MultiClusterEngines.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	Describes rules that define the resource.	object
spec *required*	MultiClusterEngineSpec defines the desired state of MultiClusterEngine.	See *List of specs*

11.5.3.2. List of specs

Name	Description	Schema
nodeSelector *optional*	Set the nodeselectors.	map[string]string
imagePullSecret *optional*	Override pull secret for accessing MultiClusterEngine operand and endpoint images.	string

Name	Description	Schema
tolerations *optional*	Tolerations causes all components to tolerate any taints.	[]corev1.Toleration
targetNamespace *optional*	Location where MCE resources will be placed.	string

11.6. PLACEMENTS API (V1ALPHA1)

11.6.1. Overview

This documentation is for the Placement resource for multicluster engine for Kubernetes. Placement resource has four possible requests: create, query, delete and update.

11.6.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.6.1.2. Tags

- cluster.open-cluster-management.io : Create and manage Placements

11.6.2. Paths

11.6.2.1. Query all Placements

```
GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements
```

11.6.2.1.1. Description

Query your Placements for more details.

11.6.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

11.6.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.6.2.1.4. Consumes

- **placement/yaml**

11.6.2.1.5. Tags

- cluster.open-cluster-management.io

11.6.2.2. Create a Placement

```
POST /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements
```

11.6.2.2.1. Description

Create a Placement.

11.6.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the placement to be created.	Placement

11.6.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content

HTTP Code	Description	Schema
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.6.2.2.4. Consumes

- **placement/yaml**

11.6.2.2.5. Tags

- cluster.open-cluster-management.io

11.6.2.2.6. Example HTTP request

11.6.2.2.6.1. Request body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1alpha1",
  "kind" : "Placement",
  "metadata" : {
    "name" : "placement1",
    "namespace": "ns1"
  },
  "spec": {
    "predicates": [
      {
        "requiredClusterSelector": {
          "labelSelector": {
            "matchLabels": {
              "vendor": "OpenShift"
            }
          }
        }
      }
    ]
  },
  "status" : { }
}
```

11.6.2.3. Query a single Placement

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/placements/{placement_name}
```

11.6.2.3.1. Description

Query a single Placement for more details.

11.6.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**placement_name** *required*	Name of the Placement that you want to query.	string

11.6.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.6.2.3.4. Tags

- cluster.open-cluster-management.io

11.6.2.4. Delete a Placement

```
DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements/{placement_name}
```

11.6.2.4.1. Description

Delete a single Placement.

11.6.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**placement_name** *required*	Name of the Placement that you want to delete.	string

11.6.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.6.2.4.4. Tags

- cluster.open-cluster-management.io

11.6.3. Definitions

11.6.3.1. Placement

Name	Description	Schema
apiVersion *required*	The versioned schema of the Placement.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	The meta data of the Placement.	object
spec *required*	The specification of the Placement.	spec

spec

Name	Description	Schema
ClusterSets *optional*	A subset of ManagedClusterSets from which the ManagedClusters are selected. If it is empty, ManagedClusters is selected from the ManagedClusterSets that are bound to the Placement namespace. Otherwise, ManagedClusters are selected from the intersection of this subset and the ManagedClusterSets are bound to the placement namespace.	string array
numberOfClusters *optional*	The desired number of ManagedClusters to be selected.	integer (int32)
predicates *optional*	A subset of cluster predicates to select ManagedClusters. The conditional logic is *OR*.	clusterPredicate array

clusterPredicate

Name	Description	Schema
requiredClusterSelector *optional*	A cluster selector to select ManagedClusters with a label and cluster claim.	clusterSelector

clusterSelector

Name	Description	Schema
labelSelector *optional*	A selector of ManagedClusters by label.	object
claimSelector *optional*	A selector of ManagedClusters by claim.	clusterClaimSelector

clusterClaimSelector

Name	Description	Schema
matchExpressions *optional*	A subset of the cluster claim selector requirements. The conditional logic is *AND*.	< object > array

11.7. PLACEMENTDECISIONS API (V1ALPHA1)

11.7.1. Overview

This documentation is for the PlacementDecision resource for multicluster engine for Kubernetes. PlacementDecision resource has four possible requests: create, query, delete and update.

11.7.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.7.1.2. Tags

- cluster.open-cluster-management.io : Create and manage PlacementDecisions.

11.7.2. Paths

11.7.2.1. Query all PlacementDecisions

```
GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions
```

11.7.2.1.1. Description

Query your PlacementDecisions for more details.

11.7.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

11.7.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.7.2.1.4. Consumes

- **placementdecision/yaml**

11.7.2.1.5. Tags

- cluster.open-cluster-management.io

11.7.2.2. Create a PlacementDecision

```
POST /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions
```

11.7.2.2.1. Description

Create a PlacementDecision.

11.7.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the PlacementDecision to be created.	PlacementDecision

11.7.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.7.2.2.4. Consumes

- **placementdecision/yaml**

11.7.2.2.5. Tags

- cluster.open-cluster-management.io

11.7.2.2.6. Example HTTP request

11.7.2.2.6.1. Request body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1alpha1",
  "kind" : "PlacementDecision",
  "metadata" : {
    "labels" : {
      "cluster.open-cluster-management.io/placement" : "placement1"
    },
    "name" : "placement1-decision1",
    "namespace": "ns1"
  },
  "status" : { }
}
```

11.7.2.3. Query a single PlacementDecision

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

11.7.2.3.1. Description

Query a single PlacementDecision for more details.

11.7.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**placementdecision_name** *required*	Name of the PlacementDecision that you want to query.	string

11.7.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content

HTTP Code	Description	Schema
503	Service unavailable	No Content

11.7.2.3.4. Tags

- cluster.open-cluster-management.io

11.7.2.4. Delete a PlacementDecision

```
DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

11.7.2.4.1. Description

Delete a single PlacementDecision.

11.7.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**placementdecision_name** *required*	Name of the PlacementDecision that you want to delete.	string

11.7.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.7.2.4.4. Tags

- cluster.open-cluster-management.io

11.7.3. Definitions

11.7.3.1. PlacementDecision

Name	Description	Schema
apiVersion *required*	The versioned schema of PlacementDecision.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	The meta data of PlacementDecision.	object

11.8. MANAGED SERVICE ACCOUNT (TECHNOLOGY PREVIEW)

11.8.1. Overview

This documentation is for the **ManagedServiceAccount** resource for the multicluster engine for Kubernetes operator. The **ManagedServiceAccount** resource has four possible requests: create, query, delete, and update.

11.8.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.8.1.2. Tags

- **managedserviceaccounts.multicluster.openshift.io`** : Create and manage **ManagedServiceAccounts**

11.8.2. Paths

11.8.2.1. Create a ManagedServiceAccount

```
POST /apis/multicluster.openshift.io/v1alpha1/ManagedServiceAccounts
```

11.8.2.1.1. Description

Create a **ManagedServiceAccount**.

11.8.2.1.2. Parameters

Type	Name	Description	Schema

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the ManagedServiceAccount to be created.	ManagedServiceAccount

11.8.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.8.2.1.4. Consumes

- **managedserviceaccount/yaml**

11.8.2.1.5. Tags

- managedserviceaccount.multicluster.openshift.io

11.8.2.1.5.1. Request body


```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "managedserviceaccount.authentication.open-cluster-management.io"
  },
  "spec": {
    "group": "authentication.open-cluster-management.io",
    "names": {
      "kind": "ManagedServiceAccount",
      "listKind": "ManagedServiceAccountList",
      "plural": "managedserviceaccounts",
```

```
      "singular": "managedserviceaccount"
    },
    "scope": "Namespaced",
    "versions": [
     {
       "name": "v1alpha1",
       "schema": {
         "openAPIV3Schema": {
           "description": "ManagedServiceAccount is the Schema for the
managedserviceaccounts\nAPI",
           "properties": {
             "apiVersion": {
               "description": "APIVersion defines the versioned schema of this representation\nof an
object. Servers should convert recognized schemas to the latest\ninternal value, and may reject
unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#resources",
               "type": "string"
             },
             "kind": {
               "description": "Kind is a string value representing the REST resource this\nobject
represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be
updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#types-kinds",
               "type": "string"
             },
             "metadata": {
               "type": "object"
             },
             "spec": {
               "description": "ManagedServiceAccountSpec defines the desired state of
ManagedServiceAccount",
               "properties": {
                 "rotation": {
                   "description": "Rotation is the policy for rotation the credentials.",
                   "properties": {
                     "enabled": {
                       "default": true,
                       "description": "Enabled prescribes whether the ServiceAccount token\nwill be rotated
from the upstream",
                       "type": "boolean"
                     },
                     "validity": {
                       "default": "8640h0m0s",
                       "description": "Validity is the duration for which the signed ServiceAccount\ntoken is
valid.",
                       "type": "string"
                     }
                   },
                   "type": "object"
                 },
                 "ttlSecondsAfterCreation": {
                   "description": "ttlSecondsAfterCreation limits the lifetime of a
ManagedServiceAccount.\nIf the ttlSecondsAfterCreation field is set, the
ManagedServiceAccount\nwill be automatically deleted regardless of the
ManagedServiceAccount's\nstatus. When the ManagedServiceAccount is deleted, its
lifecycle\nguarantees (e.g. finalizers) will be honored. If this field is unset,\nthe
```

ManagedServiceAccount won't be automatically deleted. If this\nfield is set to zero, the ManagedServiceAccount becomes eligible\nfor deletion immediately after its creation. In order to use ttlSecondsAfterCreation,\nthe EphemeralIdentity feature gate must be enabled.",
"exclusiveMinimum": true,
"format": "int32",
"minimum": 0,
"type": "integer"
}
},
"required": [
"rotation"
],
"type": "object"
},
"status": {
"description": "ManagedServiceAccountStatus defines the observed state of\nManagedServiceAccount",
"properties": {
"conditions": {
"description": "Conditions is the condition list.",
"items": {
"description": "Condition contains details for one aspect of the current\nstate of this API Resource. --- This struct is intended for direct\nuse as an array at the field path .status.conditions. For example,\ntype FooStatus struct{ // Represents the observations of a\nfoo's current state. // Known .status.conditions.type are:\n\"Available\", \"Progressing\", and \"Degraded\" // +patchMergeKey=type\n // +patchStrategy=merge // +listType=map // +listMapKey=type\n Conditions []metav1.Condition `json:\"conditions,omitempty\"\npatchStrategy:\"merge\" patchMergeKey:\"type\" protobuf:\"bytes,1,rep,name=conditions\"`\n\n // other fields }",
"properties": {
"lastTransitionTime": {
"description": "lastTransitionTime is the last time the condition\ntransitioned from one status to another. This should be when\nthe underlying condition changed. If that is not known, then\nusing the time when the API field changed is acceptable.",
"format": "date-time",
"type": "string"
},
"message": {
"description": "message is a human readable message indicating\ndetails about the transition. This may be an empty string.",
"maxLength": 32768,
"type": "string"
},
"observedGeneration": {
"description": "observedGeneration represents the .metadata.generation\nthat the condition was set based upon. For instance, if .metadata.generation\nis currently 12, but the .status.conditions[x].observedGeneration\nis 9, the condition is out of date with respect to the current\nstate of the instance.",
"format": "int64",
"minimum": 0,
"type": "integer"
},
"reason": {
"description": "reason contains a programmatic identifier indicating\nthe reason for the condition's last transition. Producers\nof specific condition types may define expected values and\nmeanings for this field, and whether the values are considered\na guaranteed API. The value should be a CamelCase string.\nThis field may not be empty.",

```
                    "maxLength": 1024,
                    "minLength": 1,
                    "pattern": "^[A-Za-z]([A-Za-z0-9_,:]*[A-Za-z0-9_])?$",
                    "type": "string"
                   },
                   "status": {
                    "description": "status of the condition, one of True, False, Unknown.",
                    "enum": [
                     "True",
                     "False",
                     "Unknown"
                    ],
                    "type": "string"
                   },
                   "type": {
                    "description": "type of condition in CamelCase or in foo.example.com/CamelCase.\n---
Many .condition.type values are consistent across resources\nlike Available, but because arbitrary
conditions can be useful\n(see .node.status.conditions), the ability to deconflict is\nimportant. The
regex it matches is (dns1123SubdomainFmt/)?(qualifiedNameFmt)",
                    "maxLength": 316,
                    "pattern": "^([a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*/)?(([A-Za-z0-
9][-A-Za-z0-9_.]*)?[A-Za-z0-9])$",
                    "type": "string"
                   }
                  },
                  "required": [
                   "lastTransitionTime",
                   "message",
                   "reason",
                   "status",
                   "type"
                  ],
                  "type": "object"
                 },
                 "type": "array"
                },
                "expirationTimestamp": {
                 "description": "ExpirationTimestamp is the time when the token will expire.",
                 "format": "date-time",
                 "type": "string"
                },
                "tokenSecretRef": {
                 "description": "TokenSecretRef is a reference to the corresponding
ServiceAccount's\nSecret, which stores the CA certficate and token from the managed\ncluster.",
                 "properties": {
                  "lastRefreshTimestamp": {
                   "description": "LastRefreshTimestamp is the timestamp indicating\nwhen the token in
the Secret is refreshed.",
                   "format": "date-time",
                   "type": "string"
                  },
                  "name": {
                   "description": "Name is the name of the referenced secret.",
                   "type": "string"
                  }
                 },
```

```
                "required": [
                  "lastRefreshTimestamp",
                  "name"
                ],
                "type": "object"
              }
            },
            "type": "object"
          }
        },
        "type": "object"
      }
    },
    "served": true,
    "storage": true,
    "subresources": {
      "status": {}
    }
  }
 ]
},
"status": {
 "acceptedNames": {
  "kind": "",
  "plural": ""
 },
 "conditions": [],
 "storedVersions": []
}
}
```


11.8.2.2. Query a single ManagedServiceAccount

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccoun
t_name}
```

11.8.2.2.1. Description

Query a single **ManagedServiceAccount** for more details.

11.8.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**managedserviceaccount_name** *required*	Name of the **ManagedServiceAccount** that you want to query.	string

11.8.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.8.2.2.4. Tags

- cluster.open-cluster-management.io

11.8.2.3. Delete a **ManagedServiceAccount**

```
DELETE /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccoun
t_name}
```

11.8.2.3.1. Description

Delete a single **ManagedServiceAccount**.

11.8.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**managedserviceaccount_name** *required*	Name of the **ManagedServiceAccount** that you want to delete.	string

11.8.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content

HTTP Code	Description	Schema
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

11.8.2.3.4. Tags

- cluster.open-cluster-management.io

11.8.3. Definitions

11.8.3.1. ManagedServiceAccount

Name	Description	Schema
apiVersion *required*	The versioned schema of the **ManagedServiceAccount**.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	The meta data of the **ManagedServiceAccount**.	object
spec *required*	The specification of the **ManagedServiceAccount**.	

CHAPTER 12. UNINSTALLING

When you uninstall multicluster engine for Kubernetes, you see two different levels of the process: A *custom resource removal* and a *complete operator uninstall*. It might take up to five minutes to complete the uninstall process.

- The custom resource removal is the most basic type of uninstall that removes the custom resource of the **MultiClusterEngine** instance but leaves other required operator resources. This level of uninstall is helpful if you plan to reinstall using the same settings and components.
- The second level is a more complete uninstall that removes most operator components, excluding components such as custom resource definitions. When you continue with this step, it removes all of the components and subscriptions that were not removed with the custom resource removal. After this uninstall, you must reinstall the operator before reinstalling the custom resource.

12.1. PREREQUISITE: DETACH ENABLED SERVICES

Before you uninstall the multicluster engine for Kubernetes engine, you must detach all of the clusters that are managed by that engine. To avoid errors, detach all clusters that are still managed by the engine, then try to uninstall again.

- If you have managed clusters attached, you might see the following message.

  ```
  Cannot delete MultiClusterEngine resource because ManagedCluster resource(s) exist
  ```

  For more information about detaching clusters, see the *Removing a cluster from management* section by selecting the information for your provider in Creating a cluster.

12.2. REMOVING RESOURCES BY USING COMMANDS

1. If you have not already. ensure that your OpenShift Container Platform CLI is configured to run **oc** commands. See Getting started with the OpenShift CLI in the OpenShift Container Platform documentation for more information about how to configure the **oc** commands.
2. Change to your project namespace by entering the following command. Replace *namespace* with the name of your project namespace:

   ```
   oc project <namespace>
   ```

3. Enter the following command to remove the **MultiClusterEngine** custom resource:

   ```
   oc delete multiclusterengine --all
   ```

   You can view the progress by entering the following command:

   ```
   oc get multiclusterengine -o yaml
   ```

4. Enter the following commands to delete the multicluster-engine **ClusterServiceVersion** in the namespace it is installed in:

```
❯ oc get csv
NAME                 DISPLAY                       VERSION   REPLACES   PHASE
```

```
multicluster-engine.v2.0.0   multicluster engine for Kubernetes   2.0.0          Succeeded

❯ oc delete clusterserviceversion multicluster-engine.v2.0.0
❯ oc delete sub multicluster-engine
```

The CSV version shown here may be different.

12.3. DELETING THE COMPONENTS BY USING THE CONSOLE

When you use the RedHat OpenShift Container Platform console to uninstall, you remove the operator. Complete the following steps to uninstall by using the console:

1. In the OpenShift Container Platform console navigation, select **Operators** > **Installed Operators** > **multicluster engine for Kubernetes**.
2. Remove the **MultiClusterEngine** custom resource.
   a. Select the tab for *Multiclusterengine*.
   b. Select the *Options* menu for the MultiClusterEngine custom resource.
   c. Select **Delete MultiClusterEngine**.
3. Run the clean-up script according to the procedure in the following section.
   Tip: If you plan to reinstall the same multicluster engine for Kubernetes version, you can skip the rest of the steps in this procedure and reinstall the custom resource.
4. Navigate to **Installed Operators**.
5. Remove the _ multicluster engine for Kubernetes_ operator by selecting the *Options* menu and selecting **Uninstall operator**.

12.4. TROUBLESHOOTING UNINSTALL

If the multicluster engine custom resource is not being removed, remove any potential remaining artifacts by running the clean-up script.

a. Copy the following script into a file:

```
#!/bin/bash
oc delete apiservice v1.admission.cluster.open-cluster-management.io
v1.admission.work.open-cluster-management.io
oc delete validatingwebhookconfiguration multiclusterengines.multicluster.openshift.io
oc delete mce --all
```

CHAPTER 13. RUNNING THE MUST-GATHER COMMAND TO TROUBLESHOOT

To get started with troubleshooting, learn about the troubleshooting scenarios for users to run the **must-gather** command to debug the issues, then see the procedures to start using the command.

Required access: Cluster administrator

13.1. MUST-GATHER SCENARIOS

- **Scenario one:** Use the *Documented troubleshooting* section to see if a solution to your problem is documented. The guide is organized by the major functions of the product.
  With this scenario, you check the guide to see if your solution is in the documentation.
- **Scenario two:** If your problem is not documented with steps to resolve, run the **must-gather** command and use the output to debug the issue.
- **Scenario three:** If you cannot debug the issue using your output from the **must-gather** command, then share your output with Red Hat Support.

13.2. MUST-GATHER PROCEDURE

See the following procedure to start using the **must-gather** command:

1. Learn about the **must-gather** command and install the prerequisites that you need at Gathering data about your cluster in the RedHat OpenShift Container Platform documentation.
2. Log in to your cluster. For the usual use-case, you should run the **must-gather** while you are logged into your *engine* cluster.
   Note: If you want to check your managed clusters, find the **gather-managed.log** file that is located in the the **cluster-scoped-resources** directory:

   ```
   <your-directory>/cluster-scoped-resources/gather-managed.log>
   ```

   Check for managed clusters that are not set **True** for the JOINED and AVAILABLE column. You can run the **must-gather** command on those clusters that are not connected with **True** status.
3. Add the multicluster engine for Kubernetes image that is used for gathering data and the directory. Run the following command, where you insert the image and the directory for the output:

   ```
   oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.0.0 --dest-dir=<directory>
   ```
4. Go to your specified directory to see your output, which is organized in the following levels:
   - Two peer levels: **cluster-scoped-resources** and **namespace** resources.
   - Sub-level for each: API group for the custom resource definitions for both cluster-scope and namespace-scoped resources.
   - Next level for each: YAML file sorted by **kind**.

13.3. MUST-GATHER IN A DISCONNECTED ENVIRONMENT

Complete the following steps to run the **must-gather** command in a disconnected environment:

1. In a disconnected environment, mirror the Red Hat operator catalog images into their mirror registry. For more information, see Install on disconnected networks.

2. Run the following command to extract logs, which reference the image from their mirror registry:

```
REGISTRY=registry.example.com:5000
IMAGE=$REGISTRY/multicluster-engine/must-gather-rhel8@sha256:ff9f37eb400dc1f7d07a9b6f2da9064992934b69847d17f59e385783c071b9d8

oc adm must-gather --image=$IMAGE --dest-dir=./data
```

You can open a bug for the product team here.