Red Hat Advanced Cluster Management for Kubernetes 2.5

多集群引擎

了解如何使用多集群引擎操作器。

Last Updated: 2023-07-15

Red Hat Advanced Cluster Management for Kubernetes 2.5 多集群引擎

了解如何使用多集群引擎操作器。

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

了解如何使用多集群引擎。

目录

第 1 章 要求和建议

在安装 Kubernetes operator 1.0 的多集群引擎前，请查看以下系统配置要求和设置：

- 支持的操作系统和平台
- 网络配置

重要：您必须在没有安装早于 2.5 版本的 Red Hat Advanced Cluster Management for Kubernetes 的集群上安装 Kubernetes 的多集群引擎。Kubernetes 的多集群引擎无法在 2.5 之前的版本上与 Red Hat Advanced Cluster Management for Kubernetes 共存，因为它们提供一些相同的管理组件。建议您在之前没有安装 Red Hat Advanced Cluster Management 的集群中安装 Kubernetes 的多集群引擎。如果您在版本 2.5.0 或更高版本中使用 Red Hat Advanced Cluster Management for Kubernetes，那么 Kubernetes 的多集群引擎将已安装在集群中。

1.1. 用于 KUBERNETES 集群和受管集群的多集群引擎支持的操作系统和平台

支持的操作系统：

平台	Kubernetes 集群的多集群引擎支持	支持受管集群
Red Hat OpenShift Container Platform 3.11.200, 及以后的 3.11.x 版本	否	是
Red Hat OpenShift Container Platform 4.8.2 及更新的版本	是	是
Red Hat OpenShift Container Platform on Amazon Web Services	是	是
Red Hat OpenShift Container Platform on Microsoft Azure	是	是
Red Hat OpenShift Container Platform on Google Cloud Platform	是	是
Red Hat OpenShift Kubernetes Engine	否	是
Google Kubernetes Engine(Google GKE)（Kubernetes 1.17 及更高版本）	否	是
Amazon Elastic Kubernetes Service(Amazon EKS)（Kubernetes 1.17.6 及更高版本）	否	是

Microsoft Azure Kubernetes Service(Microsoft AKS)（Kubernetes 1.19.6 及更高版本）	否	是

1.2. 网络配置

将您的网络设置配置为允许以下的连接：

1.2.1. Kubernetes operator 集群网络要求的多集群引擎

有关 Kubernetes 集群网络要求的多集群引擎，请查看下表：

方向	连接	端口（如果指定）
出站	云供应商 API	
出站	置备的受管集群的 Kubernetes API 服务器	6443
出站和入站	受管集群上的 **WorkManager** 服务路由	443
入站	来自受管集群的 Kubernetes 集群的多集群引擎的 Kubernetes API 服务器	6443
入站	从 GitHub 到 Kubernetes 集群的多集群引擎的 post-commit hook。只有在使用特定应用程序管理功能时才需要此设置。	6443

1.2.2. 受管集群网络要求

有关受管集群网络要求，请查看下表：

方向	连接	端口（如果指定）
出站和入站	Kubernetes 集群的多集群引擎的 Kubernetes API 服务器	6443

第 2 章 开始使用

2.1. 简介

在为 Kubernetes operator 安装多集群引擎前，请参阅 系统配置要求和设置。在集群中安装并运行了受支持的 OpenShift Container Platform 版本后，您可以继续在线安装。

2.2. 创建和管理集群

安装后，您已准备好创建、导入和管理集群。从 Kubernetes 集群的多集群引擎中，您可以创建其他 Openshift Container Platform 集群来管理。

1. 请参阅创建集群以了解您可以创建的受管集群的类型。
2. 如果您有一个需要手动导入的集群，可以参阅导入集群以了解如何导入受管集群。
3. 当不再需要管理集群时，您可以查看分离受管集群。

2.3. MANIFESTWORK 示例

您可以参阅 Deploying workload with ManifestWork 中的过程和示例。

第 3 章 在线安装

Kubernetes operator 的多集群引擎使用 Operator Lifecycle Manager 安装，用于管理安装、升级和删除包含 Kubernetes 引擎的多集群引擎的组件。

需要的访问权限：集群管理员

重要：

- 您必须在之前没有安装早于 Red Hat Advanced Cluster Management for Kubernetes 2.5 的集群上安装 Kubernetes 的多集群引擎。Kubernetes 的多集群引擎无法在 2.5 之前的版本上与 Red Hat Advanced Cluster Management for Kubernetes 共存，因为它们提供一些相同的管理组件。建议您在之前没有安装 Red Hat Advanced Cluster Management 的集群中安装 Kubernetes 的多集群引擎。如果您在版本 2.5.0 或更高版本中使用 Red Hat Advanced Cluster Management for Kubernetes，那么 Kubernetes 的多集群引擎将已安装在集群中。

- 对于 OpenShift Container Platform Dedicated 环境，必须具有 **cluster-admin** 权限。默认情况下，**dedicated-admin** 角色没有在 OpenShift Container Platform Dedicated 环境中创建命名空间所需的权限。

- 默认情况下，引擎组件会在 OpenShift Container Platform 集群的 worker 节点上安装，而无需额外的配置。您可以使用 OpenShift Container Platform OperatorHub Web 控制台界面或使用 OpenShift Container Platform CLI 将引擎安装到 worker 节点上。

- 如果您使用带有基础架构节点的 OpenShift Container Platform 集群配置，则可以使用带有额外资源参数的 OpenShift Container Platform CLI 将引擎安装到这些基础架构节点上。不是所有引擎组件都支持基础架构节点，因此在基础架构节点上为 Kubernetes 安装多集群引擎时，仍需要一些 worker 节点。详情请参阅*在基础架构节点上安装 Kubernetes 引擎*部分。

- 如果您计划导入不是由 OpenShift Container Platform 或 Kubernetes 多集群引擎创建的 Kubernetes 集群，则需要配置镜像 pull secret。有关如何配置镜像 pull secret 和其他高级配置的详情，请参考本文档的 Advanced configuration 部分中的选项。

  - 先决条件
  - 确认 OpenShift Container Platform 安装
  - 从 OperatorHub Web 控制台界面安装
  - 通过 OpenShift Container Platform CLI 安装
  - 在基础架构节点上安装

3.1. 先决条件

在为 Kubernetes 安装多集群引擎前，请查看以下要求：

- 您的 Red Hat OpenShift Container Platform 集群必须通过 OpenShift Container Platform 控制台在 OperatorHub 目录中访问适用于 Kubernetes operator 的多集群引擎。

- 您需要访问 catalog.redhat.com。

- OpenShift Container Platform 版本 4.8 或更高版本必须部署到您的环境中，且必须通过 OpenShift Container Platform CLI 登录。如需 OpenShift Container Platform，请参阅以下安装文档：

  - OpenShift Container Platform 版本 4.10

- OpenShift Container Platform 版本 4.9
- OpenShift Container Platform 版本 4.8

- 您的 OpenShift Container Platform 命令行界面（CLI）被配置为运行 **oc** 命令。如需有关安装和配置 OpenShift Container Platform CLI 的信息，请参阅 CLI 入门。
- OpenShift Container Platform 权限必须允许创建命名空间。
- 需要有一个互联网连接来访问 Operator 的依赖项。
- 要在 OpenShift Container Platform Dedicated 环境中安装，请参阅以下内容：
  - 您必须已配置并运行了 OpenShift Container Platform Dedicated 环境。
  - 您必须在要安装引擎的 OpenShift Container Platform Dedicated 环境中具有 **cluster-admin** 授权。

3.2. 确认 OPENSHIFT CONTAINER PLATFORM 安装

您必须有一个受支持的 OpenShift Container Platform 版本，包括 registry 和存储服务，并可以正常工作。有关安装 OpenShift Container Platform 的更多信息，请参阅 OpenShift Container Platform 文档。

1. 验证 OpenShift Container Platform 集群中还没有安装 Kubernetes engine operator 的多集群引擎。Kubernetes operator 的多集群引擎在每个 OpenShift Container Platform 集群中只允许一个安装。如果没有安装，请继续执行以下步骤。
2. 要确保正确设置 OpenShift Container Platform 集群，请使用以下命令访问 OpenShift Container Platform Web 控制台：

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

   请参见以下示例输出：

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

3. 在浏览器中打开 URL 并检查结果。如果控制台 URL 显示 **console-openshift-console.router.default.svc.cluster.local**，当安装 OpenShift Container Platform 时把 **openshift_master_default_subdomain** 设置为这个值。请参阅以下 URL 示例 **：https://console-openshift-console.apps.new-coral.purple-chesterfield.com**。

您可以继续为 Kubernetes 安装多集群引擎。

3.3. 从 OPERATORHUB WEB 控制台界面安装

最佳实践： 从 OpenShift Container Platform 导航中的 *Administrator* 视图，安装 OpenShift Container Platform 提供的 OperatorHub Web 控制台界面。

1. 选择 **Operators** > **OperatorHub** 来访问可用 operator 列表，选择 *multicluster engine for Kubernetes* operator。
2. 点 **Install**。
3. 在 *Operator 安装*页面中，选择安装选项：

- Namespace：
  - Kubernetes 引擎的多集群引擎必须安装在自己的命名空间或项目中。
  - 默认情况下，OperatorHub 控制台安装过程会创建一个名为 **multicluster-engine** 的命名空间。**最佳实践：** 继续使用 **multicluster-engine** 命名空间（如果可用）。
  - 如果已存在名为 **multicluster-engine** 的命名空间，请选择不同的命名空间。
- Channel：选择与要安装的发行版本相对应的频道。当您选择频道时，它会安装指定的发行版本，并确定以后获得该发行版本中的勘误更新。
- Approval strategy：批准策略指定了用户需要如何处理应用到您的频道或发行版本的更新。
  - 选择 **Automatic** （默认选择）以确保会自动应用该发行版本中的任何更新。
  - 选择 **Manual** 在有更新可用时接收通知。如果您对更新的应用有疑问，这可能是您的最佳实践。

注： 要升级到下一个次版本，您必须返回到 *OperatorHub* 页面，并为更当前的发行版本选择一个新频道。

4. 选择 **Install** 以应用您的更改并创建 Operator。
5. 请参阅以下流程来创建 *MultiClusterEngine* 自定义资源。
   a. 在 OpenShift Container Platform 控制台导航中，选择 **Installed Operators** > **multicluster engine for Kubernetes**。
   b. 选择 **MultiCluster Engine** 选项卡。
   c. 选择 **Create MultiClusterEngine**。
   d. 更新 YAML 文件中的默认值。请参阅文档中的 *MultiClusterEngine 高级配置*部分中的选项。
      - 以下示例显示了您可以复制到编辑器中的默认模板：

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

6. 选择 **Create** 来初始化自定义资源。多集群引擎最多可能需要 10 分钟才能构建并启动 Kubernetes 引擎。
   创建 *MultiClusterEngine* 资源后，在 *MultiCluster Engine* 标签页中资源的状态为 **Available**。

3.4. 通过 OPENSHIFT CONTAINER PLATFORM CLI 安装

1. 为包含 Operator 要求的 Kubernetes 引擎命名空间创建一个多集群引擎。运行以下命令，其中 **namespace** 是 Kubernetes 引擎命名空间的多集群引擎的名称。在 OpenShift Container Platform 环境中，**namespace** 的值可能被称为 *Project（项目）*。

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

2. 将项目命名空间切换到您创建的命名空间。使用在第 1 步中创建的 Kubernetes 引擎命名空间的多集群引擎名称替换 **namespace**。

```
oc project <namespace>
```

3. 创建 YAML 文件来配置 **OperatorGroup** 资源。每个命名空间只能有一个 operator 组。将 **default** 替换为 operator 组的名称。将 **namespace** 替换为项目命名空间的名称。请参见以下示例：

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

4. 运行以下命令来创建 **OperatorGroup** 资源。将 **operator-group** 替换为您创建的 operator 组 YAML 文件的名称：

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

5. 创建 YAML 文件来配置 OpenShift Container Platform 订阅。文件内容应类似以下示例：

```
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
```

注：要在基础架构节点上安装 Kubernetes 引擎的多集群引擎，请参阅 Operator Lifecycle Manager 订阅其他配置部分。

6. 运行以下命令来创建 OpenShift Container Platform 订阅。使用您创建的订阅文件的名称替换 **subscription**：

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

7. 创建 YAML 文件来配置 **MultiClusterEngine** 自定义资源。您的默认模板应类似以下示例：

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

注：要在基础架构节点上安装 Kubernetes 引擎的多集群引擎，请参阅 MultiClusterEngine 自定义资源额外配置部分：

8. 运行以下命令来创建 **MultiClusterEngine** 自定义资源。将 **custom-resource** 替换为自定义资源文件的名称：

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

如果此步骤失败并显示以下错误，则仍然会创建并应用这些资源。创建资源后几分钟内再次运行命令：

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

9. 运行以下命令来获取自定义资源。在运行以下命令后，在 **status.phase** 字段中显示 **MultiClusterEngine** 自定义资源状态 **Available** 可能需要最多 10 分钟时间：

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

如果您重新安装 Kubernetes operator 的多集群引擎且 pod 没有启动，请参阅故障排除重新安装失败以了解解决这个问题的步骤。

备注：

- 具有 **ClusterRoleBinding** 的 **ServiceAccount** 会自动向多集群引擎授予 Kubernetes 多集群引擎的权限，以及有权访问为 Kubernetes 安装多集群引擎的命名空间的任何用户凭证。

3.5. 在基础架构节点上安装

OpenShift Container Platform 集群可以配置为包含用于运行批准的管理组件的基础架构节点。在基础架构节点上运行组件可避免为运行这些管理组件的节点分配 OpenShift Container Platform 订阅配额。

将基础架构节点添加到 OpenShift Container Platform 集群后，请按照 OpenShift Container Platform CLI 指令安装，并将以下配置添加到 Operator Lifecycle Manager 订阅和 **MultiClusterEngine** 自定义资源。

3.5.1. 将基础架构节点添加到 OpenShift Container Platform 集群

按照 OpenShift Container Platform 文档中的 创建基础架构机器集 中所述的步骤进行操作。基础架构节点配置有 Kubernetes **污点（taint）**和标签（**label**），以便防止非管理工作负载在它们上运行。

要与 Kubernetes 多集群引擎提供的基础架构节点启用兼容，请确保您的基础架构节点应用了以下 **taint** 和 **label**：

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

3.5.2. Operator Lifecycle Manager Subscription 额外配置

在应用 Operator Lifecycle Manager 订阅前，添加以下配置：

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

3.5.3. MultiClusterEngine 自定义资源额外配置

在应用 **MultiClusterEngine** 自定义资源前添加以下附加配置：

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

第 4 章 在断开连接的网络中安装

您可能需要在没有连接到互联网的 Red Hat OpenShift Container Platform 集群上安装 multicluster engine operator。在断开连接的引擎中安装的步骤需要一些与连接安装相同的步骤。

重要：您必须在没有安装早于 2.5 版本的 Red Hat Advanced Cluster Management for Kubernetes 的集群上安装 Kubernetes 的多集群引擎。Kubernetes 的多集群引擎无法在 2.5 之前的版本上与 Red Hat Advanced Cluster Management for Kubernetes 共存，因为它们提供一些相同的管理组件。建议您在之前没有安装 Red Hat Advanced Cluster Management 的集群中安装 Kubernetes 的多集群引擎。如果您在版本 2.5.0 或更高版本中使用 Red Hat Advanced Cluster Management for Kubernetes，那么 Kubernetes 的多集群引擎将已安装在集群中。

您必须下载软件包副本以在安装过程中访问它们，而不是在安装过程中直接从网络访问它们。

- 先决条件
- 确认 OpenShift Container Platform 安装
- 准备在基础架构节点上安装引擎

4.1. 先决条件

在安装 multicluster engine operator 前，您必须满足以下要求：

- Red Hat OpenShift Container Platform 版本 4.8 或更高版本必须部署到您的环境中，且必须使用 CLI 登录。
- 您需要访问 catalog.redhat.com。
  注： 要管理裸机集群，您必须使用 OpenShift Container Platform 版本 4.8 或更高版本。

  请参阅 OpenShift Container Platform 版本 4.10、OpenShift Container Platform 版本 4.8。
- 您的 Red Hat OpenShift Container Platform CLI 需要版本 4.8 或更高版本，并配置为运行 **oc** 命令。如需有关安装和配置 Red Hat OpenShift CLI 的信息，请参阅 CLI 入门。
- 您的 Red Hat OpenShift Container Platform 权限必须允许创建命名空间。
- 必须有一 个有互联网连接的工作站来下载 operator 的依赖软件包。

4.2. 确认 OPENSHIFT CONTAINER PLATFORM 安装

- 您必须有一个受支持的 OpenShift Container Platform 版本，包括 registry 和存储服务，在集群中安装并正常工作。如需有关 OpenShift Container Platform 版本 4.8 的信息，请参阅 OpenShift Container Platform 文档。
- 连接后，您可以确保正确设置了 OpenShift Container Platform 集群。访问 OpenShift Container Platform Web 控制台。
  运行 **kubectl -n openshift-console get route** 命令来访问 OpenShift Container Platform Web 控制台。请参见以下示例输出：

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

  本例中的控制台 URL 为 **https:// console-openshift-console.apps.new-coral.purple-chesterfield.com**。在浏览器中打开 URL 并检查结果。

如果控制台 URL 显示 **console-openshift-console.router.default.svc.cluster.local**，当安装 OpenShift Container Platform 时把 **openshift_master_default_subdomain** 设置为这个值。

4.3. 在断开连接的环境中安装

重要： 您需要将所需的镜像下载到镜像 registry 中，以便在断开连接的环境中安装 Operator。如果没有下载，您可能会在部署过程中收到 **ImagePullBackOff** 错误。

按照以下步骤在断开连接的环境中为 Kubernetes Operator 安装多集群引擎：

1. 创建镜像 registry。如果您还没有镜像 registry，请按照 Red Hat OpenShift Container Platform 文档中的为断开连接的环境创建镜像的容器镜像的步骤来创建。
   如果已有镜像 registry，可以配置和使用现有 registry。

2. **注：** 对于裸机，您需要在 **install-config.yaml** 文件中为断开连接的 registry 提供证书信息。要访问受保护的断开连接的 registry 中的镜像，您必须提供证书信息，以便 Kubernetes operator 的多集群引擎可以访问 registry。

   a. 复制 registry 中的证书信息。

   b. 在编辑器中打开 **install-config.yaml** 文件。

   c. 找到 **additionalTrustBundle: |** 条目。

   d. 在 **additionalTrustBundle** 行后添加证书信息。内容应类似以下示例：

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

3. **重要：** 如果需要以下监管策略，则需要额外的镜像 registry：

   - Container Security Operator 策略：镜像位于源 **registry.redhat.io/quay** 中。
   - Compliance operator 策略：镜像位于源 **registry.redhat.io/compliance** 中
   - Gatekeeper operator 策略：镜像位于源 **registry.redhat.io/rhacm2** 中
     参阅以下所有三个 operator 的镜像列表示例：

   ```
   - 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. 保存 **install-config.yaml** 文件。

5. 创建一个包含 **ImageContentSourcePolicy** 的 YAML 文件，其名称为 **rhacm-policy.yaml**。**注：** 如果您在正在运行的集群中修改此操作，则会导致所有节点的滚动重启。

```
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. 输入以下命令应用 ImageContentSourcePolicy 文件：

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

7. 启用断开连接的 Operator Lifecycle Manager Red Hat Operator 和 Community Operator。Kubernetes operator 的多集群引擎包含在 Operator Lifecycle Manager Red Hat Operator 目录中。

8. 为 Red Hat Operator 目录配置离线 Operator Lifecycle Manager。按照 Red Hat OpenShift Container Platform 文档中 受限网络部分中使用 Operator Lifecycle Manager 中的步骤操作。

9. 现在，您已在断开连接的 Operator Lifecycle Manager 中已有镜像，从 Operator Lifecycle Manager 目录继续为 Kubernetes operator 安装多集群引擎。

如需了解所需步骤，请参阅在线安装。

第 5 章 控制台概述

OpenShift Container Platform 控制台插件包括在 OpenShift Container Platform 4.10 web 控制台中，并可集成。要使用这个功能，必须启用控制台插件。Multicluster engine operator 在 **Infrastructure** 和 **Credentials** 导航项中显示某些控制台功能。如果安装了 Red Hat Advanced Cluster Management，您会看到更多的控制台功能。

注： 对于启用插件的 OpenShift Container Platform 4.10，您可以从 OpenShift Container Platform 控制台下拉菜单中选择 **All Clusters** 来访问 OpenShift Container Platform 控制台中的 Red Hat Advanced Cluster Management。

1. 要禁用插件，请确保处于 OpenShift Container Platform 控制台的 *Administrator* 视角中。
2. 在导航中找到 **Administration**，再点 **Cluster Settings**，然后点 *Configuration* 选项卡。
3. 从 *Configuration resources* 列表中，点带有 **operator.openshift.io** API 组的 **Console** 资源，其中包含 web 控制台的集群范围配置。
4. 点 *Console 插件* 选项卡。**mce** 插件被列出。**注：** 如果安装了 Red Hat Advanced Cluster Management，它也会被列为 **acm**。
5. 从表中修改插件状态。几分钟后，会提示您输入刷新控制台。

第 6 章 高级配置

Kubernetes operator 的多集群引擎使用部署所有所需组件的 operator 安装。在安装过程中或安装后，可以通过向 **MultiClusterEngine** 自定义资源添加一个或多个属性来进一步配置 Kubernetes operator 的多集群引擎：

6.1. 自定义镜像 PULL SECRET

如果您计划导入不是由 OpenShift Container Platform 或 Kubernetes operator 多集群引擎创建的 Kubernetes 集群，生成一个包含 OpenShift Container Platform pull secret 信息的 secret，以便从发布 registry 中访问授权内容。

OpenShift Container Platform 集群的 secret 要求由 Kubernetes 的 OpenShift Container Platform 和多集群引擎自动解决，因此如果您没有导入其他类型的 Kubernetes 集群，则不必创建 secret。

重要： 这些 secret 是特定于命名空间的，因此请确保处于用于引擎的命名空间中。

1. 选择 **Download pull secret** 从 cloud.redhat.com/openshift/install/pull-secret 下载 OpenShift Container Platform pull secret 文件。您的 OpenShift Container Platform pull secret 与您的 Red Hat Customer Portal ID 相关联，在所有 Kubernetes 供应商中都是相同的。
2. 运行以下命令来创建 secret:

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

   - 将 **secret** 替换为您要创建的 secret 的名称。
   - 将 **namespace** 替换为项目命名空间，因为 secret 是特定于命名空间的。
   - 将 **path-to-pull-secret** 替换为您下载的 OpenShift Container Platform pull secret 的路径。

以下示例显示，如果使用自定义 pull secret，要使用的 **spec.imagePullSecret** 模板。将 **secret** 替换为 pull secret 的名称：

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  imagePullSecret: <secret>
```

6.2. 目标命名空间

可通过在 **MultiClusterEngine** 自定义资源中指定位置，在指定的命名空间中安装操作对象。此命名空间在 **MultiClusterEngine** 自定义资源的应用程序上创建。

重要： 如果没有指定目标命名空间，Operator 将安装到 **multicluster-engine** 命名空间，并在 **MultiClusterEngine** 自定义资源规格中设置它。

以下示例显示了可以用来指定目标命名空间的 **spec.targetNamespace** 模板。使用目标命名空间的名称替换 **target**。**注：** **target** 目标命名空间不能是 **default** 命名空间：

```
apiVersion: operator.open-cluster-management.io/v1
```

```
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  targetNamespace: <target>
```

6.3. AVAILABILITYCONFIG

Red Hat Advanced Cluster Management hub 集群有两个可用功能： **High** 和 **Basic**。默认情况下，hub 集群的可用性为 **High**，hub 集群组件**副本数** 为 **2**。它提供了对故障转移功能的支持，但消耗的资源数量比可用性为 **Basic**（**副本数**为**1**） 的集群多。

以下示例显示了具有 **Basic** 可用性的 **spec.availabilityConfig** 模板：

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  availabilityConfig: "Basic"
```

6.4. NODESELECTOR

您可以在 **MultiClusterEngine** 中定义一组节点选择器，以安装到集群中的特定节点。以下示例显示了将 Red Hat Advanced Cluster Management pod 分配给带有标签 **node-role.kubernetes.io/infra** 的节点的 **spec.nodeSelector** ：

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

6.5. 容限（TOLERATIONS）

您可以定义容限列表，以允许 **MultiClusterEngine** 容许在集群中定义的特定污点。以下示例显示了与 **node-role.kubernetes.io/infra** 污点匹配的 **spec.tolerations** ：

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

默认情况下，以上 infra-node 容限在 pod 上设置，而不在配置中指定任何容限。在配置中自定义容限将替换此默认行为。

6.6. MANAGEDSERVICEACCOUNT 附加组件（技术预览）

默认情况下，**Managed-ServiceAccount** 附加组件被禁用。启用该组件后，您可以在受管集群上创建或删除服务帐户。要在启用此附加组件的环境中安装，请在 **spec.overrides** 的 **MultiClusterEngine** 规格中包括以下内容：

```
apiVersion: operator.open-cluster-management.io/v1
```

```
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: managedserviceaccount-preview
      enabled: true
```

在创建 **MultiClusterEngine** 后，可以在命令行中编辑资源并将 **managedserviceaccount-preview** 组件设置为 **enabled: true** 来启用 **Managed-ServiceAccount** 插件。或者，您可以运行以下命令，将 <multiclusterengine-name> 替换为 **MultiClusterEngine** 资源的名称。

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

请参阅启用 ManagedServiceAccount 附加组件 以启用。

6.7. HYPERSHIFT 附加组件（技术预览）

默认情况下，**Hypershift** 附加组件被禁用。要在启用此附加组件的环境中安装，请在 **spec.overrides** 的 **MultiClusterEngine** 值中包含以下内容：

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: hypershift-preview
      enabled: true
```

在创建 **MultiClusterEngine** 后，可在命令行中编辑资源，将 **hypershift-preview** 组件设置为 **enabled: true** 来启用 **Hypershift** 附加组件。或者，您可以运行以下命令，将 <multiclusterengine-name> 替换为 **MultiClusterEngine** 资源的名称：

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

第 7 章 启用 MANAGEDSERVICEACCOUNT 附加组件（技术预览）

当您为 Kubernetes operator 安装多集群引擎时，**managedServiceAccount** add-on 会被默认禁用。启用该组件后，您可以在受管集群上创建或删除服务帐户。

需要的访问权限： Editor

当在 hub 集群的 **<managed_cluster>** 命名空间中创建了一个 **ManagedServiceAccount** 自定义资源后，会在受管集群中创建一个 **ServiceAccount**。

TokenRequest 由受管集群中的 **ServiceAccount** 组成，指向受管集群的 Kubernetes API 服务器。然后，令牌将存储在 hub 集群上的 **<target_managed_cluster>** 命名空间中的 **Secret** 中。

注： 令牌可以过期并可以被轮转。有关令牌请求的更多信息，请参阅 TokenRequest。

7.1. 先决条件

- Red Hat OpenShift Container Platform 版本 4.9 或更高版本必须部署到您的环境中，且必须使用 CLI 登录。
- 您需要安装了 multicluster engine for Kubernetes operator。

7.2. 启用 MANAGEDSERVICEACCOUNT

要为 hub 集群和受管集群启用 **Managed-ServiceAccount** 附加组件，请完成以下步骤：

1. 在 hub 集群上启用 **ManagedServiceAccount** 附加组件。请参阅高级配置以了解更多信息。
2. 部署 **ManagedServiceAccount** 附加组件，并将其应用到目标受管集群。创建以下 YAML 文件，并将 **target_managed_cluster** 替换为您要应用 **Managed-ServiceAccount** 附加组件的受管集群的名称：

```
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. 运行以下命令以应用该文件：

```
oc apply -f -
```

   您已为受管集群启用了 **Managed-ServiceAccount** 插件。请参阅以下步骤来配置 **ManagedServiceAccount**。

4. 使用以下 YAML 源创建 **ManagedServiceAccount** 自定义资源：

```
apiVersion: authentication.open-cluster-management.io/v1alpha1
kind: ManagedServiceAccount
metadata:
  name: <managed_serviceaccount_name>
```

```
  namespace: <target_managed_cluster>
spec:
  rotation: {}
```

- 将 **managed_serviceaccount_name** 替换为 **ManagedServiceAccount** 的名称。
- 将 **target_managed_cluster** 替换为您要将 **ManagedServiceAccount** 应用到的受管集群的名称。

5. 要验证，请查看 **ManagedServiceAccount** 对象状态中的 **tokenSecretRef** 属性，以查找 secret 名称和命名空间。使用您的帐户和集群名称运行以下命令：

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

6. 查看包含连接到受管集群中创建的 **ServiceAccount** 的已检索令牌的 **Secret**。运行以下命令：

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

第 8 章 创建集群

Kubernetes 的多集群引擎使用内部 Hive 组件创建 Red Hat OpenShift Container Platform 集群。请参阅以下信息以了解如何创建集群。

- 使用 ClusterDeployment 创建集群
- 使用集群池创建集群

8.1. 使用 CLUSTERDEPLOYMENT 创建集群

ClusterDeployment 是一个 Hive 自定义资源。请参阅以下文档以了解如何创建单个集群：

按照使用 Hive 文档创建 **ClusterDeployment** 自定义资源。

8.2. 使用 CLUSTERPOOL 创建集群

ClusterPool 也是用于创建多个集群的 Hive 自定义资源。使用 Hive **ClusterPool** API 创建集群。

按照集群池文档置备集群。

第 9 章 导入集群

在为 Kubernetes operator 安装多集群引擎后，就可以导入集群来管理。使用 Red Hat OpenShift Container Platform CLI，您可以使用您要导入的集群的 **kubeconfig** 文件导入集群。另外，您还可以在导入的集群中手动运行导入命令。本文档中提供了这两个程序。

参阅以下流程从 CLI 导入：

9.1. 先决条件

- 您可以使用 Linux（x86_64、s390x、ppc64le）或 macOS。
- 确保为 Kubernetes operator 安装多集群引擎，并在 Kubernetes 集群上安装了 MultiClusterEngine 自定义资源。
- 您需要一个要管理的另一个单独集群，以及互联网连接。
- 您需要 OpenShift Container Platform CLI 版本 4.8 或更高版本来运行 **oc** 命令。如需有关安装和配置 Red Hat OpenShift CLI **oc** 的信息，请参阅 OpenShift CLI 入门。
  注： 从 OpenShift Container Platform 控制台下载 CLI 工具的安装文件。
- 如果您要导入不是由 OpenShift Container Platform 创建的集群，则需要定义的 **multiclusterengine.spec.imagePullSecret**。安装 Kubernetes 的多集群引擎时可能已经创建了此 secret。有关定义 secret 的更多信息，请参阅高级配置。

9.2. 准备导入

1. 登录到您的*引擎集群*。*引擎（engine）集群*是一个集群，其中包含 Kubernetes operator 和自定义资源的多集群引擎。运行以下命令：

   ```
   oc login
   ```

2. 在 engine 集群中运行以下命令，以创建项目：
   注： 在 **CLUSTER_NAME** 中定义的集群名称，它也用作 **.yaml** 文件和命令中的集群命名空间：

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

3. 运行以下命令来创建名称空间：

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

4. 使用以下 YAML 示例编辑示例 **ManagedCluster** ：

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

5. **可选：** 在本发行版本中，您*无法*自动导入您的 hub 集群作为一个受管集群，称为 **local-cluster**。要手动使受管集群变为 **local-cluster**，请添加 **metadata.labels.local-cluster: "true"**。请参见以下 YAML 示例，并确保名称为 **local-cluster**。如果 **local-cluster** 不是名称，导入将失败或创建未预期的结果：

```
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. 将文件保存为 **managed-cluster.yaml**。

7. 使用以下命令应用 YAML 文件：

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

9.3. 使用自动导入 SECRET 导入

在仍登录到您的引擎集群时继续执行以下步骤：

1. 检索您要导入的集群的 **kubeconfig** 文件，或者您要导入的集群的 kube API 服务器和令牌。请参阅 Kubernetes 集群的文档，以了解在什么位置找到 **kubeconfig** 文件或 kube api 服务器和令牌

2. 使用您的 **kubeconfig** 或您的 server/token 对来创建包含类似以下模板的 YAML 文件：

```
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. 将文件保存为 auto-import-secret.yaml

4. 使用您要导入的集群的 **kubeconfig** 文件，在 **${CLUSTER_NAME}** 命名空间中生成导入 secret。使用 **kubeconfig** 和 **CLUSTER_NAME** 的路径运行以下命令：

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

注： 自动导入 secret 只使用一次，在导入过程完成后会被删除。

5. 验证您的导入集群的 **JOINED** 和 **AVAILABLE** 状态。从 Kubernetes 集群的多集群引擎运行以下命令：

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

在您要导入的独立集群中继续执行以下步骤：

6. 登录到您要导入的集群。运行以下命令：

```
oc login
```

7. 验证您要导入的集群中的 pod 状态。运行以下命令：

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

8. 导入的集群将安装附加组件是 **AVAILABLE**。验证集群中附加组件的 Pod 状态。运行以下命令：

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

集群现已导入。

9.4. 使用手动命令导入

重要： 导入命令包含复制到每个导入集群的 pull secret 信息。具有访问导入集群权限的所有用户都可以查看 pull secret 信息。

1. 获取由引擎集群上的导入控制器生成的 **klusterlet-crd.yaml**。运行以下命令：

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

2. 获取由引擎集群上的导入控制器生成的 **import.yaml**。运行以下命令：

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

在您要导入的独立集群中继续执行以下步骤：

3. 登录到您要导入的集群。

```
oc login
```

4. 应用您在上一步中生成的 **klusterlet-crd.yaml**。运行以下命令：

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

5. 应用之前生成的 **import.yaml** 文件。运行以下命令：

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

6. 验证您要导入的集群中的 pod 状态。运行以下命令：

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

7. 验证您要导入的集群的 **JOINED** 和 **AVAILABLE** 状态。在引擎集群中运行以下命令：

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

附加组件安装在您导入的集群后为 **AVAILABLE**。

8. 验证您要导入的集群中的附加组件的 Pod 状态。运行以下命令：

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

现在，集群已导入，您可以从引擎集群中管理该集群。

9.5. 分离受管集群

受管集群是一个成功导入的集群。要从引擎集群分离受管集群，请运行以下命令：

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

您的集群现已分离。

第 10 章 使用清单工作部署工作负载

您可以从 Kubernetes 集群的多集群引擎将工作负载部署到受管集群上。例如：请参阅以下带有 **ManifestWork** 的示例，从您的 Kubernetes 集群的多集群引擎在受管集群中创建基本的部署：

1. 登录到 Kubernetes 集群的多集群引擎：

```
oc login
```

2. 创建一个 YAML 文件来配置 **ManifestWork** 资源，如下例所示。将 **CLUSTER_NAME** 替换为从导入集群文档导入的受管集群的名称。应用文件时，YAML 示例部署到受管集群 **default** 命名空间：

```
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. 应用 YAML 文件。运行以下命令：

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

4. 运行以下命令，从 Kubernetes 集群的多集群引擎检查 manifest **Work** 的状态：

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

5. 登录到受管集群以查看结果。使用以下命令：

```
oc login
```

6. 查看您使用 Kubernetes 集群的多集群引擎创建的部署：

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

您可以使用以下命令查看创建的 pod：

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

如果查看所创建的 pod 的日志，您会看到类似如下的消息：

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

第 11 章 API

您可以使用 Kubernetes operator 的多集群引擎来访问集群生命周期管理的 API。**用户需要的访问权限：**您只能执行已分配角色的操作。如需更多信息，请参阅以下每个资源的 API 文档：

- Clusters API
- ClusterSets API(v1beta1)
- Clusterview API
- ClusterSetBindings API (v1beta1)
- MultiClusterEngine API
- Placements API (v1alpha1)
- PlacementDecisions API (v1alpha1)
- 管理的服务帐户（技术预览）

11.1. CLUSTERS API

11.1.1. 概述

本文档介绍了与 Kubernetes 的多集群引擎的集群资源相关的 API 信息。集群资源有 4 个可用的请求：create、query、delete 和 update。

11.1.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.1.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.1.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.1.1.4. Tags

- cluster.open-cluster-management.io：创建和管理集群

11.1.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.1.2. 路径

11.1.2.1. 查询所有集群

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

11.1.2.1.1. 描述

查询集群以获取更多详细信息。

11.1.2.1.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串

11.1.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.1.2.1.4. 使用

- **cluster/yaml**

11.1.2.1.5. Tags

- cluster.open-cluster-management.io

11.1.2.2. 创建集群

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

11.1.2.2.1. 描述

创建集群

11.1.2.2.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
Body	**body** *必需*	描述要创建集群的参数。	Cluster

11.1.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.1.2.2.4. 使用

- **cluster/yaml**

11.1.2.2.5. Tags

- cluster.open-cluster-management.io

11.1.2.2.6. HTTP 请求示例

11.1.2.2.6.1. 请求正文

```
{
  "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. 查询单个集群

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

11.1.2.3.1. 描述

查询单个集群以获取更多详细信息。

11.1.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**cluster_name** *必需*	要查询的集群的名称。	字符串

11.1.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.1.2.3.4. Tags

- cluster.open-cluster-management.io

11.1.2.4. 删除集群

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

11.1.2.4.1. 描述

删除单个集群

11.1.2.4.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**cluster_name** *必需*	要删除的集群的名称。	字符串

11.1.2.4.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.1.2.4.4. Tags

- cluster.open-cluster-management.io

11.1.3. 定义

11.1.3.1. Cluster

Name	模式
apiVersion *必需*	字符串
kind *必需*	字符串
metadata *必需*	对象

Name	模式
spec *必需*	spec

spec

Name	模式
hubAcceptsClient *必需*	bool
managedClusterClientConfigs *可选*	< managedClusterClientConfigs > array
leaseDurationSeconds *可选*	integer (int32)

managedClusterClientConfigs

Name	描述	模式
URL *必需*		字符串
CABundle *可选*	Pattern : **"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"**	字符串（字节）

11.2. CLUSTERSETS API（V1ALPHA1）

11.2.1. 概述

本文档介绍了与 Kubernetes 的多集群引擎的 Clusterset 资源相关的 API 信息。Clusterset 资源有 4 个可能的请求：create、query、delete 和 update。

11.2.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.2.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.2.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.2.1.4. Tags

- cluster.open-cluster-management.io：创建和管理 Clustersets

11.2.1.5. 外部文档

描述：查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.2.2. 路径

11.2.2.1. 查询所有集群集（clusterset）

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

11.2.2.1.1. 描述

查询 Clustersets 以获取更多详细信息。

11.2.2.1.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串

11.2.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.2.2.1.4. 使用

- **clusterset/yaml**

11.2.2.1.5. Tags

- cluster.open-cluster-management.io

11.2.2.2. 创建一个 clusterset

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

11.2.2.2.1. 描述

创建 Clusterset。

11.2.2.2.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
Body	**body** *必需*	描述要创建的 clusterset 的参数。	Clusterset

11.2.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.2.2.2.4. 使用

- **clusterset/yaml**

11.2.2.2.5. Tags

- cluster.open-cluster-management.io

11.2.2.2.6. HTTP 请求示例

11.2.2.2.6.1. 请求正文

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

11.2.2.3. 查询单个集群集

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

11.2.2.3.1. 描述

查询单个集群集以获取更多详细信息。

11.2.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**clusterset_name** *必需*	要查询的集群集的名称。	字符串

11.2.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.2.2.3.4. Tags

- cluster.open-cluster-management.io

11.2.2.4. 删除集群集

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

11.2.2.4.1. 描述

删除单个集群集。

11.2.2.4.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**clusterset_name** *必需*	要删除的集群集的名称。	字符串

11.2.2.4.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.2.2.4.4. Tags

- cluster.open-cluster-management.io

11.2.3. 定义

11.2.3.1. Clusterset

Name	模式
apiVersion *必需*	字符串

Name	模式
kind *必需*	字符串
metadata *必需*	对象

11.3. CLUSTERVIEW API (V1ALPHA1)

11.3.1. 概述

本文档介绍了与 Kubernetes 的多集群引擎的 **clusterview** 资源相关的 API 信息。**clusterview** 资源提供了一个 CLI 命令，可让您查看您可以访问的受管集群和受管集群集的列表。三个可能的请求有：list、get 和 watch。

11.3.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.3.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.3.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.3.1.4. Tags

- clusterview.open-cluster-management.io：查看 ID 可访问的受管集群列表。

11.3.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.3.2. 路径

11.3.2.1. 获取受管集群

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

11.3.2.1.1. 描述

查看您可以访问的受管集群列表。

11.3.2.1.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串

11.3.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.3.2.1.4. 使用

- **managedcluster/yaml**

11.3.2.1.5. Tags

- clusterview.open-cluster-management.io

11.3.2.2. 列出受管集群

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

11.3.2.2.1. 描述

查看您可以访问的受管集群列表。

11.3.2.2.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串

类型	Name	描述	模式
Body	body *可选*	要列出受管集群的用户 ID 的名称。	字符串

11.3.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.3.2.2.4. 使用

- **managedcluster/yaml**

11.3.2.2.5. Tags

- clusterview.open-cluster-management.io

11.3.2.2.6. HTTP 请求示例

11.3.2.2.6.1. 请求正文

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

11.3.2.3. 观察受管集群集

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

11.3.2.3.1. 描述

观察您可以访问的受管集群。

11.3.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**clusterview_name** *可选*	要监视的用户 ID 的名称。	字符串

11.3.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.3.2.4. 列出受管集群集。

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

11.3.2.4.1. 描述

列出您可以访问的受管集群。

11.3.2.4.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串

类型	Name	描述	模式
路径	**clusterview_name** *可选*	要监视的用户 ID 的名称。	字符串

11.3.2.4.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.3.2.5. 列出受管集群集。

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

11.3.2.5.1. 描述

列出您可以访问的受管集群。

11.3.2.5.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**clusterview_name** *可选*	要监视的用户 ID 的名称。	字符串

11.3.2.5.3. 响应

HTTP 代码	描述	模式
200	成功	无内容

HTTP 代码	描述	模式
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.3.2.6. 观察受管集群集。

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

11.3.2.6.1. 描述

观察您可以访问的受管集群。

11.3.2.6.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**clusterview_name** *可选*	要监视的用户 ID 的名称。	字符串

11.3.2.6.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.4. CLUSTERSETBINDINGS API (V1ALPHA1)

11.4.1. 概述

本文档介绍了与 Kubernetes 的多集群引擎的 clustersetbinding 资源相关的 API 信息。Clustersetbinding 资源有 4 个可能的请求：create、query、delete 和 update。

11.4.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.4.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.4.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.4.1.4. Tags

- cluster.open-cluster-management.io：创建和管理 clustersetbindings

11.4.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.4.2. 路径

11.4.2.1. 查询所有 clustersetbindings

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

11.4.2.1.1. 描述

查询 clustersetbindings 以获取更多详细信息。

11.4.2.1.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串

类型	Name	描述	模式
路径	**namespace** *必需*	要使用的命名空间，如 default。	字符串

11.4.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.4.2.1.4. 使用

- **clustersetbinding/yaml**

11.4.2.1.5. Tags

- cluster.open-cluster-management.io

11.4.2.2. 创建 clustersetbinding

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

11.4.2.2.1. 描述

创建 clustersetbinding。

11.4.2.2.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}；ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**namespace** *必需*	要使用的命名空间，如 default。	字符串

类型	Name	描述	模式
Body	body *必需*	描述要创建的 clustersetbinding 的参数。	Clustersetbinding

11.4.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.4.2.2.4. 使用

- **clustersetbinding/yaml**

11.4.2.2.5. Tags

- cluster.open-cluster-management.io

11.4.2.2.6. HTTP 请求示例

11.4.2.2.6.1. 请求正文

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

11.4.2.3. 查询单个 clustersetbinding

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

11.4.2.3.1. 描述

查询单个 clustersetbinding 获取更多详细信息。

11.4.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**namespace** *必需*	要使用的命名空间，如 default。	字符串
路径	**clustersetbinding_name** *必需*	要查询的 clustersetbinding 的名称。	字符串

11.4.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.4.2.3.4. Tags

- cluster.open-cluster-management.io

11.4.2.4. 删除 clustersetbinding

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

11.4.2.4.1. 描述

删除单个 clustersetbinding。

11.4.2.4.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**namespace** *必需*	要使用的命名空间，如 default。	字符串
路径	**clustersetbinding_name** *必需*	要删除的 clustersetbinding 的名称。	字符串

11.4.2.4.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.4.2.4.4. Tags

- cluster.open-cluster-management.io

11.4.3. 定义

11.4.3.1. Clustersetbinding

Name	模式
apiVersion *必需*	字符串
kind *必需*	字符串

Name	模式
metadata *必需*	对象
spec *必需*	spec

spec

Name	模式
clusterSet *必需*	字符串

11.5. API

11.5.1. 概述

本文档介绍了与 Kubernetes 的多集群引擎的 MultiClusterEngine 资源相关的 API 信息。**MultiClusterEngine** 资源有 4 个可用的请求：create、query、delete 和 update。

11.5.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.5.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.5.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.5.1.4. Tags

- multiclusterengines.multicluster.openshift.io : 创建和管理 MultiClusterEngines

11.5.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.5.2. 路径

11.5.2.1. 创建 MultiClusterEngine

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

11.5.2.1.1. 描述

创建一个 MultiClusterEngine。

11.5.2.1.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
Body	**body** *必需*	描述要创建的 MultiClusterEngine 的参数。	MultiClusterEngine

11.5.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.5.2.1.4. 使用

- **MultiClusterEngines/yaml**

11.5.2.1.5. Tags

- multiclusterengines.multicluster.openshift.io

11.5.2.1.5.1. 请求正文

```
{
  "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. 查询所有 MultiClusterEngines

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

11.5.2.2.1. 描述

查询多集群引擎以获取更多详细信息。

11.5.2.2.2. 参数

类型	Name	描述	模式
Header	COOKIE *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串

11.5.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.5.2.2.4. 使用

- **operator/yaml**

11.5.2.2.5. Tags

- multiclusterengines.multicluster.openshift.io

11.5.2.3. 删除 MultiClusterEngine Operator

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

11.5.2.3.1. 参数

类型	Name	描述	模式

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**name** *必需*	要删除的 multiclusterengine 的名称。	字符串

11.5.2.3.2. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.5.2.3.3. Tags

- multiclusterengines.multicluster.openshift.io

11.5.3. 定义

11.5.3.1. MultiClusterEngine

名称	描述	模式
apiVersion *必需*	版本化的 MultiClusterEngine 模式。	字符串
kind *必需*	代表 REST 资源的字符串值。	字符串
metadata *必需*	描述定义资源的规则。	对象
spec *必需*	MultiClusterEngineSpec 定义 MultiClusterEngine 的所需状态。	请参阅 *specs 列表*

11.5.3.2. specs 列表

名称	描述	模式
nodeSelector *可选*	设置 nodeselectors。	map[string]string
imagePullSecret *可选*	覆盖用于访问 MultiClusterEngine 操作对象和端点镜像的 pull secret。	字符串
tolerations *可选*	容限会导致所有组件都容许任何污点。	[]corev1.Toleration
targetNamespace *可选*	放置 MCE 资源的位置。	字符串

11.6. PLACEMENTS API (V1ALPHA1)

11.6.1. 概述

本文档介绍了用于 Kubernetes 的多集群引擎的放置资源。放置资源有 4 个可用的请求：create、query、delete 和 update。

11.6.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.6.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.6.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.6.1.4. Tags

- cluster.open-cluster-management.io：创建和管理放置

11.6.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.6.2. 路径

11.6.2.1. 查询所有放置

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

11.6.2.1.1. 描述

查询您的放置以获取更多详细信息。

11.6.2.1.2. 参数

类型	Name	描述	模式
Header	COOKIE *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串

11.6.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.6.2.1.4. 使用

- **placement/yaml**

11.6.2.1.5. Tags

- cluster.open-cluster-management.io

11.6.2.2. 创建一个放置

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

11.6.2.2.1. 描述

创建一个放置。

11.6.2.2.2. 参数

类型	Name	描述	模式
Header	COOKIE *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
Body	body *必需*	描述要创建的放置的参数。	Placement

11.6.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.6.2.2.4. 使用

- **placement/yaml**

11.6.2.2.5. Tags

- cluster.open-cluster-management.io

11.6.2.2.6. HTTP 请求示例

11.6.2.2.6.1. 请求正文

```
{
  "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. 查询单个放置

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

11.6.2.3.1. 描述

查询单个放置以获取更多详细信息。

11.6.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**placement_name** *必需*	要查询的放置名称。	字符串

11.6.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.6.2.3.4. Tags

- cluster.open-cluster-management.io

11.6.2.4. 删除一个放置

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

11.6.2.4.1. 描述

删除一个放置。

11.6.2.4.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**placement_name** *必需*	要删除的放置名称。	字符串

11.6.2.4.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.6.2.4.4. Tags

- cluster.open-cluster-management.io

11.6.3. 定义

11.6.3.1. Placement

名称	描述	模式
apiVersion *必需*	放置的版本化模式。	字符串

名称	描述	模式
kind *必需*	代表 REST 资源的字符串值。	字符串
metadata *必需*	放置的元数据。	对象
spec *必需*	放置的规格。	spec

spec

Name	描述	模式
ClusterSets *可选*	从中选择 ManagedClusterSet 的 ManagedClusterSet 子集。如果为空，则从绑定到 Placement 命名空间的 ManagedClusterSets 中选择 ManagedClusters。否则，ManagedClusters 会从这个子集的交集中选择，ManagedClusterSets 会绑定到 placement 命名空间。	字符串数组
numberOfClusters *可选*	要选择的 ManagedClusters 数量。	integer (int32)
predicates *可选*	选择 ManagedClusters 的集群 predicates 子集。条件逻辑为 *OR*。	clusterPredicate array

clusterPredicate

名称	描述	模式
requiredClusterSelector *可选*	选择带有标签和集群声明的 ManagedClusters 的集群选择器。	clusterSelector

clusterSelector

名称	描述	模式
labelSelector *可选*	按标签的 ManagedClusters 选择器。	对象

名称	描述	模式
claimSelector *可选*	按声明的 ManagedClusters 选择器。	clusterClaimSelector

clusterClaimSelector

名称	描述	模式
matchExpressions *可选*	集群声明选择器要求的子集。条件逻辑是 *AND*。	< object > 数组

11.7. PLACEMENTDECISIONS API (V1ALPHA1)

11.7.1. 概述

本文档介绍了与 Kubernetes 的多集群引擎的 PlacementDecision 资源相关的 API 信息。
PlacementDecision 资源有 4 个可用的请求：create、query、delete 和 update。

11.7.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.7.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.7.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.7.1.4. Tags

- cluster.open-cluster-management.io：创建和管理放置Decisions。

11.7.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.7.2. 路径

11.7.2.1. 查询所有 PlacementDecisions

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

11.7.2.1.1. 描述

查询您的 PlacementDecisions 获取更多详细信息。

11.7.2.1.2. 参数

类型	Name	描述	模式
Header	COOKIE *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串

11.7.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.7.2.1.4. 使用

- **placementdecision/yaml**

11.7.2.1.5. Tags

- cluster.open-cluster-management.io

11.7.2.2. 创建 PlacementDecision

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

11.7.2.2.1. 描述

创建 PlacementDecision。

11.7.2.2.2. 参数

类型	Name	描述	模式

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
Body	**body** *必需*	描述要创建的 PlacementDecision 的参数。	PlacementDecision

11.7.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.7.2.2.4. 使用

- **placementdecision/yaml**

11.7.2.2.5. Tags

- cluster.open-cluster-management.io

11.7.2.2.6. HTTP 请求示例

11.7.2.2.6.1. 请求正文

```
{
  "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. 查询单个 PlacementDecision

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

11.7.2.3.1. 描述

查询单个 PlacementDecision 获取更多详细信息。

11.7.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**placementdecision_name** *必需*	要查询的 PlacementDecision 的名称。	字符串

11.7.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.7.2.3.4. Tags

- cluster.open-cluster-management.io

11.7.2.4. 删除 PlacementDecision

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

11.7.2.4.1. 描述

删除单个 PlacementDecision。

11.7.2.4.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}； ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**placementdecision_name** *必需*	要删除的 PlacementDecision 的名称。	字符串

11.7.2.4.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.7.2.4.4. Tags

- cluster.open-cluster-management.io

11.7.3. 定义

11.7.3.1. PlacementDecision

名称	描述	模式
apiVersion *必需*	版本化的 PlacementDecision schema	字符串
kind *必需*	代表 REST 资源的字符串值。	字符串
metadata *必需*	PlacementDecision 的元数据。	对象

11.8. 管理的服务帐户（技术预览）

11.8.1. 概述

本文档介绍了与 Kubernetes operator 的多集群引擎的 **ManagedServiceAccount** 资源相关的 API 信息。**ManagedServiceAccount** 资源有 4 个可用的请求：create、query、delete 和 update。

11.8.1.1. 联系信息

Contact Email : apiteam@swagger.io

11.8.1.2. 许可证信息

License : Apache 2.0
License URL : http://www.apache.org/licenses/LICENSE-2.0.html
Terms of service : http://swagger.io/terms/

11.8.1.3. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

11.8.1.4. Tags

- **managedserviceaccounts.multicluster.openshift.io'**: 创建和管理 **ManagedServiceAccounts**

11.8.1.5. 外部文档

描述 : 查找更多有关 Swagger 的信息。
URL : http://swagger.io

11.8.2. 路径

11.8.2.1. 创建 ManagedServiceAccount

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

11.8.2.1.1. 描述

创建 **ManagedServiceAccount**。

11.8.2.1.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
Body	**body** *必需*	描述要创建的 ManagedServiceAccount 的参数。	ManagedServiceAccount

11.8.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.8.2.1.4. 使用

- **managedserviceaccount/yaml**

11.8.2.1.5. Tags

- managedserviceaccount.multicluster.openshift.io

11.8.2.1.5.1. 请求正文


```
{
  "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. 查询单个 ManagedServiceAccount

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

11.8.2.2.1. 描述

查询单个 **ManagedServiceAccount** 获取更多详细信息。

11.8.2.2.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN}；ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**managedserviceaccount_name** *必须*	要查询的 **ManagedServiceAccount** 的名称。	字符串

11.8.2.2.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容

HTTP 代码	描述	模式
500	内部服务错误	无内容
503	服务不可用	无内容

11.8.2.2.4. Tags

- cluster.open-cluster-management.io

11.8.2.3. 删除 **ManagedServiceAccount**

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

11.8.2.3.1. 描述

删除单个 **ManagedServiceAccount**。

11.8.2.3.2. 参数

类型	Name	描述	模式
Header	**COOKIE** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**managedserviceaccount_name** *必须*	要删除的 **ManagedServiceAccount** 的名称。	字符串

11.8.2.3.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容
404	未找到资源	无内容
500	内部服务错误	无内容
503	服务不可用	无内容

11.8.2.3.4. Tags

- cluster.open-cluster-management.io

11.8.3. 定义

11.8.3.1. ManagedServiceAccount

名称	描述	模式
apiVersion *必需*	**ManagedServiceAccount** 的版本化模式。	字符串
kind *必需*	代表 REST 资源的字符串值。	字符串
metadata *必需*	**ManagedServiceAccount** 的元数据。	对象
spec *必需*	**ManagedServiceAccount** 的规格。	

第 12 章 卸装

在卸载 Kubernetes 的多集群引擎时，您会看到两个不同的流程级别：*删除自定义资源*和*完成 Operator 卸载*。完成卸载过程最多可能需要五分钟。

- 删除自定义资源是最基本的卸载类型，它会删除 **MultiClusterEngine** 实例的自定义资源，但会保留其他所需的 operator 资源。如果您计划使用相同的设置和组件重新安装，这个卸载级别很有用。
- 第二个级别是更完整的卸载，可删除大多数 Operator 组件，不包括自定义资源定义等组件。当您继续执行此步骤时，它会删除所有没有通过删除自定义资源而被删除的组件和订阅。在卸载后，您必须在重新安装自定义资源前重新安装 Operator。

12.1. 先决条件：分离启用的服务

在卸载 Kubernetes 引擎的多集群引擎前，您必须分离所有由该引擎管理的集群。为了避免错误，分离仍由引擎管理的所有集群，然后尝试再次卸载。

- 如果附加了受管集群，您可能会看到以下信息。

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

  有关分离集群的更多信息，请参阅*从管理部分删除集群*，在 创建集群中选择与您的供应商相关的信息。

12.2. 使用命令删除资源

1. 如果还没有运行 oc 命令，请确保 OpenShift Container Platform CLI 配置为运行 **oc** 命令。如需有关如何配置 **oc** 命令的更多信息，请参阅 OpenShift Container Platform 文档中的 OpenShift CLI 入门。
2. 输入以下命令来更改到您的项目命名空间。将 *namespace* 替换为项目命名空间的名称：

   ```
   oc project <namespace>
   ```

3. 输入以下命令删除 **MultiClusterEngine** 自定义资源：

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

   您可以输入以下命令来查看进度：

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

4. 输入以下命令删除在其中安装的命名空间中 multicluster-engine **ClusterServiceVersion** ：

```
❯ 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
```

此处显示的 CSV 版本可能会有所不同。

12.3. 使用控制台删除组件

当使用 Red Hat OpenShift Container Platform 控制台卸载时，需要删除 operator。使用控制台完成以下步骤进行卸载：

1. 在 OpenShift Container Platform 控制台导航中，选择 **Operators** > **Installed Operators** > **multicluster engine for Kubernetes**
2. 删除 **MultiClusterEngine** 自定义资源。
   a. 选择 *Multiclusterengine* 标签页
   b. 选择 MultiClusterEngine 自定义资源的 *Options* 菜单。
   c. 选择 **Delete MultiClusterEngine**。
3. 根据以下部分中的步骤运行清理脚本。
   提示： 如果您计划为 Kubernetes 版本重新安装相同的多集群引擎，您可以跳过这个过程中的其余步骤并重新安装自定义资源。
4. 进入 **Installed Operators**。
5. 选择 *Options* 菜单并选择 **Uninstall operator** 来删除 Kubernetes_ operator 的 _ multicluster 引擎。

12.4. 卸载故障排除

如果没有删除多集群引擎自定义资源，请通过运行清理脚本删除潜在的剩余工件。

a. 将以下脚本复制到一个文件中：

```
#!/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
```

第 13 章 运行 MUST-GATHER 命令进行故障排除

要进行故障排除，参阅可以使用 **must-gather** 命令进行调试的用户情景信息，然后使用这个命令进行故障排除。

需要的访问权限：集群管理员

13.1. MUST-GATHER 情境

- **场景一：**如果您的问题已被记录，使用 *已记录的故障排除*文档部分进行解决。这个指南按照产品的主要功能进行组织。
  在这种情况下，您可以参阅本指南来查看您的问题的解决方案是否在文档中。
- **情况 2：**如果这个指南中没有与您的问题相关的内容，运行 **must-gather** 命令并使用输出来调试问题。
- **情况 3：**无法使用 **must-gather** 命令的输出结果无法帮助解决您的问题，请向红帽支持提供您的输出。

13.2. MUST-GATHER 过程

请参阅以下流程来使用 **must-gather** 命令：

1. 了解 **must-gather** 命令以及按照 Red Hat OpenShift Container Platform 文档中的 收集集群数据所需的先决条件。
2. 登录到您的集群。对于通常的用例，您应该在登录到您的 *引擎* 集群时运行 **must-gather**。
   备注：要检查您的受管集群，找到位于 **cluster-scoped-resources** 目录中的 **gather-managed.log** 文件：

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

   检查 JOINED 和 AVAILABLE 栏没有被设置为 **True** 的受管集群。您可以在这些没有以 **True** 状态连接的集群中运行 **must-gather** 命令。
3. 为用于收集数据和目录的 Kubernetes 镜像添加多集群引擎。运行以下命令，在其中提供您要插入的镜像和输出目录：

   ```
   oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.2.0 --dest-dir=<directory>
   ```

4. 进入您指定的目录查看输出。输出以以下级别进行组织：
   - 两个对等级别：**cluster-scoped-resources** 和 **namespace** 资源。
   - 每个对等级别下的子类：用于 cluster-scope 和 namespace-scoped 资源的自定义资源定义的 API 组。
   - 每个子类的下一级： 按 **kind** 进行排序的 YAML 文件。

13.3. 在断开连接的环境中的 MUST-GATHER

在断开连接的环境中，按照以下步骤运行 **must-gather** 命令：

1. 在断开连接的环境中，将 RedHat operator 目录镜像镜像（mirror）到其 mirror registry 中。如需更多信息，请参阅在断开连接的网络中安装。

2. 运行以下命令以提取日志，从其 mirror registry 中引用镜像：

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

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

您可以在此处报告一个程序错误。