



Red Hat Advanced Cluster Management for Kubernetes 2.7

集群

使用集群生命周期和多集群引擎，您可以创建和管理集群。集群生命周期可以通过 multicluster engine operator 提供。

Red Hat Advanced Cluster Management for Kubernetes 2.7 集群

使用集群生命周期和多集群引擎，您可以创建和管理集群。集群生命周期可以通过 `multicluster engine operator` 提供。

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

了解有关集群生命周期和多集群引擎 operator 的更多信息。

目录

第 1 章 带有多集群引擎 OPERATOR 的集群生命周期概述	3
1.1. 发行注记	3
1.2. 关于多集群引擎 OPERATOR 的集群生命周期	14
1.3. 安装和升级多集群引擎 OPERATOR	21
1.4. 管理凭证	42
1.5. 集群生命周期简介	56
1.6. 发现服务简介	204
1.7. 托管 CONTROL PLANE（技术预览）	209
1.8. API	250
1.9. 故障排除	300

第 1 章 带有多集群引擎 OPERATOR 的集群生命周期概述

multicluster engine operator 是集群生命周期 Operator，它为 OpenShift Container Platform 和 Red Hat Advanced Cluster Management hub 集群提供集群管理功能。在 hub 集群中，您可以创建和管理集群，也可以销毁您创建的任何集群。您还可以休眠、恢复和分离集群。从以下文档了解更多有关集群生命周期功能的信息。

信息：

- 集群通过 Hive 资源使用 OpenShift Container Platform 集群安装程序创建。请参阅 OpenShift Container Platform 文档中的有关在 [OpenShift Container Platform 安装概述](#) 中安装集群的过程的更多信息。
- 在 OpenShift Container Platform 集群中，您可以使用 multicluster engine operator 作为集群生命周期功能的独立集群管理器，或者将其用作 Red Hat Advanced Cluster Management hub 集群的一部分。
- 如果您只使用 OpenShift Container Platform，则订阅中包含该 Operator。参阅 OpenShift Container Platform 文档中的[关于 Kubernetes operator 的多集群引擎](#)。
- 如果订阅 Red Hat Advanced Cluster Management，您还会收到安装 Operator。您可以使用 Red Hat Advanced Cluster Management hub 集群创建、管理和监控其他 Kubernetes 集群。请参阅 Red Hat Advanced Cluster Management [安装文档](#)。
- 发行镜像是您创建集群时使用的 OpenShift Container Platform 版本。对于使用 Red Hat Advanced Cluster Management 创建的集群，您可以启用自动升级发行镜像。有关 Red Hat Advanced Cluster Management 中发行镜像的更多信息，请参阅 [指定发行镜像](#)。
 - [关于多集群引擎 operator 的集群生命周期](#)
 - [发行注记](#)
 - [安装多集群引擎 operator](#)
 - [管理凭证](#)
 - [集群生命周期简介](#)
 - [发现服务简介](#)
 - [托管 control plane（技术预览）](#)
 - [API](#)
 - [故障排除](#)

集群生命周期管理架构的组件包括在[集群生命周期架构](#)中。

1.1. 发行注记

了解当前版本。

已弃用：不再支持 multicluster engine operator 2.2。文档可能仍然可用，但没有任何可用的勘误或其他更新。升级到最新支持的 multicluster engine operator 版本以获取最佳结果。

- [multicluster engine operator 中的新功能](#)

- [勘误更新](#)
- [已知的与集群生命周期相关的问题](#)
- [弃用和删除](#)

如果您在当前支持的某个版本或产品文档时遇到问题，请访问 [红帽支持](#)，您可以在其中进行故障排除、查看知识库文章、与支持团队连接，或者创建一个问题单。您必须使用您的凭证登录。

您还可以访问红帽客户门户文档，[Red Hat Customer Portal FAQ](#)。

1.1.1. 使用 multicluster engine operator 的集群生命周期新功能

重要：一些功能和组件作为[技术预览](#)发布。

了解更多本发行版本的新内容：

- 如果安装了 Red Hat Advanced Cluster Management，可以通过[欢迎使用 Red Hat Advanced Cluster Management for Kubernetes](#) 了解产品概述。
- 开源的 *Open Cluster Management* 存储库可用于开源社区的交互、增长和贡献。要参与，请参阅 [open-cluster-management.io](#)。您还可以访问 [GitHub 存储库](#) 来获取更多信息。
- [安装](#)
- [集群生命周期](#)
- [托管 control plane](#)

1.1.1.1. 安装

如果安装了 OpenShift Container Platform 或 Red Hat Advanced Cluster Management，则会自动接收 multicluster engine operator。如果安装的多集群引擎 operator 有任何新功能，您可以在本节中查看它们。

1.1.1.2. 集群生命周期

了解与 multicluster engine operator 的集群生命周期有关的新新功能。

- 现在，您可以使用 **cluster-proxy-addon** 通过代理连接到受管集群中的任何服务。从 hub 直接访问受管集群，以便与某些服务交互。如需更多信息，请参阅[使用集群代理附加组件](#)。
- 现在，您可以在 Amazon Web Services GovCloud 上创建集群。如需更多信息，请参阅在 [Amazon Web Services GovCloud 上创建集群](#)。

以下 API 有新版本： - **ClusterSet**: v1beta2 - **ClusterSetBinding**: v1beta2 请参阅 [API](#) 以了解更多信息。

- 您可以使用两种 control plane 类型：hosted（托管）和 standalone（独立）。Standalone 是控制台向导功能，Hosted 提供具体步骤和指导，以便您可以通过 CLI 创建集群。
- 现在，您可以更新 **MultiClusterEngine** 自定义资源，以指定 hub 集群是否由自己管理。如需更多信息，请参阅[本地集群启用](#)。
- 您可以在断开连接的环境中创建基础架构环境时，修改 **ConfigMap** 和 **AgentServiceConfig** 文件中的设置，以指定未经身份验证的 registry。如需更多信息，请参阅[启用中央基础架构管理服务](#)。

- 现在，使用 MachinePool 扩展已正式发布，您可以轻松地配置自动扩展来扩展资源。如需更多信息，请参阅[使用 MachinePool 扩展](#)。

1.1.1.3. 托管 control plane

- **技术预览**：您可以在 Amazon Web Services 或裸机平台上置备托管 control plane 集群。如需更多信息，请参阅[托管的 control plane（技术预览）](#)。

1.1.2. 已知的与集群生命周期相关的问题

查看 multicluster engine operator 的集群生命周期的已知问题。以下列表包含本发行版本的已知问题，或从上一版本中继承的问题。对于 OpenShift Container Platform 集群，请参阅 [OpenShift Container Platform 发行注记](#)。

- [集群生命周期](#)
- [托管 control plane](#)

1.1.2.1. 集群管理

集群生命周期已知问题和限制是 multicluster engine operator 文档的集群生命周期的一部分。

1.1.2.1.1. 删除附加组件时，手动删除受管集群上所需的 VolSync CSV

当您从 hub 集群中删除 VolSync **ManagedClusterAddOn** 时，它会删除受管集群上的 VolSync operator 订阅，但不会删除集群服务版本(CSV)。要从受管集群中删除 CSV，请在您要删除 VolSync 的每个受管集群中运行以下命令：

```
oc delete csv -n openshift-operators volsync-product.v0.6.0
```

如果您安装了不同版本的 VolSync，请将 **v0.6.0** 替换为您的安装版本。

1.1.2.1.2. 删除受管集群集不会自动删除其标签

删除 **ManagedClusterSet** 后，添加到每个受管集群的标签不会被自动删除。从已删除受管集群集中包含的每个受管集群手动删除该标签。该标签类似以下示例：**cluster.open-cluster-management.io/clusterSet:<ManagedClusterSet Name>**。

1.1.2.1.3. ClusterClaim 错误

如果您针对 **ClusterPool** 创建 Hive **ClusterClaim** 并手动将 **ClusterClaimSpec** 生命周期字段设置为无效的 golang 时间值，则产品将停止实现并协调所有 **ClusterClaims**，而不仅仅是不正确的声明。

如果发生这个错误，您可以在 **clusterclaim-controller** pod 日志中看到以下内容，它是一个带有池名称和无效生命周期的特定示例：

```
E0203 07:10:38.266841      1 reflector.go:138] sigs.k8s.io/controller-runtime/pkg/cache/internal/informers_map.go:224: Failed to watch *v1.ClusterClaim: failed to list *v1.ClusterClaim: v1.ClusterClaimList.Items: [v1.ClusterClaim: v1.ClusterClaim.v1.ClusterClaim.Spec: v1.ClusterClaimSpec.Lifetime: unmarshalerDecoder: time: unknown unit "w" in duration "1w", error found in #10 byte of ...[time:"1w"},{"apiVe|..., bigger context ...|clusterPoolName":"policy-aas-hubs","lifetime":"1w"}}, {"apiVersion":"hive.openshift.io/v1","kind":"Cl|...
```

您可以删除无效的声明。

如果删除了不正确的声明，则声明可以在不需要进一步交互的情况下再次成功进行协调。

1.1.2.1.4. 产品频道与置备的集群不同步

clusterimageset 处于 **fast** 频道，但置备的集群处于 **stable** 频道。目前，产品不会将频道同步到置备的 OpenShift Container Platform 集群。

进入 OpenShift Container Platform 控制台中的正确频道。点 **Administration > Cluster Settings > Details Channel**。

1.1.2.1.5. 使用自定义 CA 证书恢复到其恢复的 hub 集群连接可能会失败

恢复受管集群使用自定义 CA 证书的 hub 集群的备份后，受管集群和 hub 集群之间的连接可能会失败。这是因为在恢复的 hub 集群上没有备份 CA 证书。要恢复连接，将受管集群的命名空间中自定义 CA 证书信息复制到恢复的 hub 集群上的 **<managed_cluster>-admin-kubeconfig** secret。

提示：如果您在创建备份副本前将此 CA 证书复制到 hub 集群，备份副本会包括 secret 信息。当使用备份副本来恢复时，hub 和受管集群之间的连接会自动完成。

1.1.2.1.6. local-cluster 可能无法自动重新创建

如果在 **disableHubSelfManagement** 被设置为 **false** 时删除 local-cluster，则 **MulticlusterHub** operator 会重新创建 local-cluster。分离 local-cluster 后，可能不会自动重新创建 local-cluster。

- 要解决这个问题，修改由 **MulticlusterHub** operator 监控的资源。请参见以下示例：

```
oc delete deployment multiclusterhub-repo -n <namespace>
```

- 要正确分离 local-cluster，在 **MultiClusterHub** 中将 **disableHubSelfManagement** 设置为 true。

1.1.2.1.7. 在创建内部集群时需要选择子网

使用控制台创建内部集群时，您必须为集群选择一个可用的子网。它没有标记为必填字段。

1.1.2.1.8. 使用 Infrastructure Operator 进行集群置备失败

当使用 Infrastructure Operator 创建 OpenShift Container Platform 集群时，ISO 镜像的文件名可能会太长。镜像名称长会导致镜像置备和集群置备失败。要确定这是否是问题，请完成以下步骤：

1. 运行以下命令，查看您要置备的集群的裸机主机信息：

```
oc get bmh -n <cluster_provisioning_namespace>
```

2. 运行 **describe** 命令以查看错误信息：

```
oc describe bmh -n <cluster_provisioning_namespace> <bmh_name>
```

3. 类似以下示例的错误表示文件名的长度问题：

Status:

Error Count: 1

Error Message: Image provisioning failed: ... [Errno 36] File name too long ...

如果出现问题，通常位于以下 OpenShift Container Platform 版本上，因为基础架构操作员不使用镜像服务：

- 4.8.17 及更早版本
- 4.9.6 及更早版本

为了避免这个错误，将 OpenShift Container Platform 升级到 4.8.18 或更高版本，或 4.9.7 或更高版本。

1.1.2.1.9. 使用不同名称重新导入后 local-cluster 状态为离线

当您意外尝试以不同名称的集群形式重新导入名为 **local-cluster** 的集群时，**local-cluster** 和重新导入的集群的状态将 离线。

要从中这个问题单中恢复，请完成以下步骤：

1. 在 hub 集群中运行以下命令，以临时编辑 hub 集群的自助管理设置：

```
oc edit mch -n open-cluster-management multiclusterhub
```

2. 添加 **spec.disableSelfManagement=true** 设置。
3. 在 hub 集群中运行以下命令以删除并重新部署 local-cluster：

```
oc delete managedcluster local-cluster
```

4. 输入以下命令删除 **local-cluster** 管理设置：

```
oc edit mch -n open-cluster-management multiclusterhub
```

5. 删除之前添加的 **spec.disableSelfManagement=true**。

1.1.2.1.10. 在代理环境中使用 Ansible 自动化进行集群置备失败

当满足以下条件时，配置为自动置备受管集群的 Automation 模板可能会失败：

- hub 集群启用了集群范围代理。
- Ansible Automation Platform 只能通过代理访问。

1.1.2.1.11. klusterlet Operator 的版本必须与 hub 集群相同

如果您通过安装 klusterlet operator 导入受管集群，klusterlet Operator 的版本必须与 hub 集群的版本相同，或者 klusterlet Operator 将无法正常工作。

1.1.2.1.12. 无法手动删除受管集群命名空间

您无法手动删除受管集群的命名空间。受管集群命名空间会在受管集群分离后自动删除。如果在分离受管集群前手动删除受管集群命名空间，受管集群会在删除受管集群后显示持续终止状态。要删除此正在终止的受管集群，请从分离的受管集群中手动删除终结器。

1.1.2.1.13. hub 集群和受管集群的时钟未同步

hub 集群和管理集群的时间可能会不同步，在控制台中显示 **unknown**，当在几分钟内会变为 **available**。确保正确配置了 OpenShift Container Platform hub 集群时间。请参阅[自定义节点](#)。

1.1.2.1.14. 不支持导入 IBM OpenShift Container Platform Kubernetes Service 集群的特定版本

您无法导入 IBM OpenShift Container Platform Kubernetes Service 版本 3.11 集群。支持 IBM OpenShift Kubernetes Service 的更新的版本。

1.1.2.1.15. 不支持为置备的集群进行自动 secret 更新

当您在云供应商一端更改云供应商访问密钥时，您还需要在 multicluster engine operator 的控制台中更新此云供应商的对应凭证。当凭证在托管受管集群的云供应商过期并尝试删除受管集群时，需要此项。

1.1.2.1.16. 无法在搜索中查看受管集群的节点信息

搜索 hub 集群中资源的 RBAC 映射。根据 RBAC 设置，用户可能无法看到来自受管集群的节点数据。搜索的结果可能与集群的 *Nodes* 页面中显示的结果不同。

1.1.2.1.17. 销毁集群的进程没有完成

当销毁受管集群时，在一小时后仍然继续显示 **Destroying** 状态，且集群不会被销毁。要解决这个问题请完成以下步骤：

1. 手动确保云中没有孤立的资源，且清理与受管集群关联的所有供应商资源。
2. 输入以下命令为正在删除的受管集群打开 **ClusterDeployment**：

```
oc edit clusterdeployment/<mycluster> -n <namespace>
```

将 *mycluster* 替换为您要销毁的受管集群的名称。

使用受管集群的命名空间替换 *namespace*。

3. 删除 **hive.openshift.io/deprovision** finalizer，以强制停止尝试清理云中的集群资源的进程。
4. 保存您的更改，验证 **ClusterDeployment** 是否已不存在。
5. 运行以下命令手动删除受管集群的命名空间：

```
oc delete ns <namespace>
```

使用受管集群的命名空间替换 *namespace*。

1.1.2.1.18. 无法使用控制台在 OpenShift Container Platform Dedicated 上升级 OpenShift Container Platform 受管集群

您不能使用 Red Hat Advanced Cluster Management 控制台升级 OpenShift Container Platform Dedicated 环境中的 OpenShift Container Platform 受管集群。

1.1.2.1.19. 工作管理器附加搜索详情

特定受管集群中特定资源的搜索详情页面可能会失败。在进行搜索前，您必须确保受管集群中的 work-manager 附加组件处于 **Available** 状态。

1.1.2.1.20. 非 Red Hat OpenShift Container Platform 受管集群必须启用 LoadBalancer

Red Hat OpenShift Container Platform 集群和非 OpenShift Container Platform 集群都支持 pod 日志功能，但非 OpenShift Container Platform 集群需要启用 **LoadBalancer** 来使用该功能。完成以下步骤以启用 **LoadBalancer**：

1. 云供应商有不同的 **LoadBalancer** 配置。有关更多信息，请访问您的云供应商文档。
2. 检查 **loggingEndpoint** 是否显示 **managedClusterInfo** 状态来验证 Red Hat Advanced Cluster Management 上是否启用了 **LoadBalancer**。
3. 运行以下命令，以检查 **loggingEndpoint.IP** 或 **loggingEndpoint.Host** 是否具有有效的 IP 地址或主机名：

```
oc get managedclusterinfo <clusterName> -n <clusterNamespace> -o json | jq -r
'.status.loggingEndpoint'
```

如需有关 **LoadBalancer** 类型的更多信息，请参阅 [Kubernetes 文档](#) 中的 *Service* 页面。

1.1.2.1.21. OpenShift Container Platform 4.10.z 不支持使用代理配置托管的 control plane 集群

当您在 OpenShift Container Platform 4.10.z 上使用集群范围代理配置创建托管服务集群时，**nodeip-configuration.service** 服务不会在 worker 节点上启动。

1.1.2.1.22. 无法在 Azure 上置备 OpenShift Container Platform 4.11 集群

因为身份验证 operator 超时错误，在 Azure 上置备 OpenShift Container Platform 4.11 集群会失败。要临时解决这个问题，在 **install-config.yaml** 文件中使用不同的 worker 节点类型，或者将 **vmNetworkingType** 参数设置为 **Basic**。请参阅以下 **install-config.yaml** 示例：

```
compute:
- hyperthreading: Enabled
  name: 'worker'
  replicas: 3
platform:
  azure:
    type: Standard_D2s_v3
    osDisk:
      diskSizeGB: 128
    vmNetworkingType: 'Basic'
```

1.1.2.1.23. 客户端无法访问 iPXE 脚本

iPXE 是开源网络引导固件。如需了解更多详细信息，请参阅 [iPXE](#)。

引导节点时，一些 DHCP 服务器中的 URL 长度限制会关闭 **InfraEnv** 自定义资源定义中的 **ipxeScript** URL，从而导致在控制台中的以下错误消息：

no bootable devices

要临时解决这个问题，请完成以下步骤：

1. 在使用辅助安装时应用 **InfraEnv** 自定义资源定义以公开 **bootArtifacts**，它可能类似以下文件：

```
status:
```

```

agentLabelSelector:
  matchLabels:
    infraenvs.agent-install.openshift.io: qe2
bootArtifacts:
  initrd: https://assisted-image-service-multicluster-engine.redhat.com/images/0000/pxe-
initrd?api_key=0000000&arch=x86_64&version=4.11
  ipxeScript: https://assisted-service-multicluster-engine.redhat.com/api/assisted-
install/v2/infra-envs/00000/downloads/files?api_key=000000000&file_name=ipxe-script
  kernel: https://mirror.openshift.com/pub/openshift-
v4/x86_64/dependencies/rhcos/4.11/latest/rhcos-live-kernel-x86_64
  rootfs: https://mirror.openshift.com/pub/openshift-
v4/x86_64/dependencies/rhcos/4.11/latest/rhcos-live-rootfs.x86_64.img

```

2. 创建代理服务器以使用短 URL 公开 **bootArtifacts**。
3. 运行以下命令复制 **bootArtifacts** 并将其添加到代理中：

```

for artifact in oc get infraenv qe2 -ojsonpath="{.status.bootArtifacts}" | jq ". | keys[]" | sed
"s/^//g"
do curl -k oc get infraenv qe2 -ojsonpath="{.status.bootArtifacts.${artifact}}" -o $artifact

```

4. 将 **ipxeScript** 工件代理 URL 添加到 **libvirt.xml** 中的 **bootp** 参数。

1.1.2.1.24. 升级 Red Hat Advanced Cluster Management 后无法删除 *ClusterDeployment*

如果您在 Red Hat Advanced Cluster Management 2.6 中使用已删除的 BareMetalAssets API，则在升级到 Red Hat Advanced Cluster Management 2.7 后无法删除 **ClusterDeployment**，因为 BareMetalAssets API 绑定到 **ClusterDeployment**。

要临时解决这个问题，请在升级到 Red Hat Advanced Cluster Management 2.7 前运行以下命令来删除 **finalizers**：

```

oc patch clusterdeployment <clusterdeployment-name> -p '{"metadata":{"finalizers":null}}' --
type=merge

```

1.1.2.1.25. 使用中央基础架构管理服务在断开连接的环境中部署的集群可能无法安装

当使用中央基础架构管理服务在断开连接的环境中部署集群时，集群节点可能无法开始安装。

这是因为集群使用从 OpenShift Container Platform 版本 4.12.0 到 4.12.2 提供的 Red Hat Enterprise Linux CoreOS live ISO 镜像创建的发现 ISO 镜像。该镜像包含一个限制性 **/etc/containers/policy.json** 文件，该文件需要来自 **registry.redhat.io** 和 **registry.access.redhat.com** 的镜像的签名。在断开连接的环境中，mirror 的镜像可能没有 mirror 的签名，这会导致发现集群节点的镜像拉取失败。Agent 镜像无法与集群节点连接，这会导致与辅助服务的通信失败。

要临时解决这个问题，将 ignition 覆盖应用到集群，将 **/etc/containers/policy.json** 文件设置为 **unrestrictive**。ignition 覆盖可以在 **InfraEnv** 自定义资源定义中设置。以下示例显示了带有覆盖的 **InfraEnv** 自定义资源定义：

```

apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: cluster
  namespace: cluster

```


1.1.3. 勘误更新

对于多集群引擎 operator，勘误更新会在发布时自动应用。

重要：为了参考，[勘误](#)链接和 GitHub 号可能会添加到内容中并在内部使用。用户可能不能使用访问的链接。

FIPS 注意：如果您没有在 `spec.ingress.sslCiphers` 中指定自己的密码，则 `multiclusterhub-operator` 会提供默认密码列表。对于 2.4，这个列表包括两个未被 FIPS 批准的加密方式。如果您从 2.4.x 或更早版本升级并希望符合 FIPS 合规性，请从 `multiclusterhub` 资源中删除以下两个加密方式：`ECD HE-ECDSA-CHACHA20-POLY1305` 和 `ECDHE-RSA-CHACHA20-POLY1305`。

1.1.3.1. Errata 2.2.13

- 为一个或多个产品容器镜像提供更新。

1.1.3.2. Errata 2.2.12

- 为一个或多个产品容器镜像提供更新。

1.1.3.3. Errata 2.2.11

- 为一个或多个产品容器镜像提供更新。
- 修复了受管集群使用已弃用的 region 和 zone 标签的问题。([ACM-8844](#))
- 修复了在 *Topology* 视图中错误显示节点区域和区域的问题。([ACM-9004](#))

1.1.3.4. Errata 2.2.10

- 为一个或多个产品容器镜像提供更新。

1.1.3.5. Errata 2.2.9

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.3.6. Errata 2.2.8

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.3.7. Errata 2.2.7

- 为一个或多个产品容器镜像和安全修复提供更新。
- 修复了在将节点添加到不属于 Hive `MachinePool` 的受管集群时导致控制台显示不正确的扩展警报的问题。([ACM-5169](#))

1.1.3.8. Errata 2.2.6

- 修复了当 secret 总文件大小太大时导致 klusterlet 代理失败的问题。([ACM-5873](#))

1.1.3.9. Errata 2.2.5

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.3.10. Errata 2.2.4

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.3.11. Errata 2.2.3

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.3.12. Errata 2.2.2

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.3.13. 勘误 2.2.1

- 为一个或多个产品容器镜像和安全修复提供更新。

1.1.4. 弃用和删除

了解产品何时被弃用或从多集群引擎 operator 中删除。考虑 *推荐操作* 中的备选操作和详细信息，它们显示在当前版本的表中和之前两个版本。

1.1.4.1. API 弃用和删除

multicluster engine operator 遵循 API 的 Kubernetes 弃用指南。有关该策略的更多详细信息，请参阅 [Kubernetes Deprecation Policy](#)。多集群引擎 operator API 仅在以下时间表外已弃用或删除：

- 所有 **V1** API 已正式发布（GA），提供 12 个月或跨三个发行版本（以更长的时间为准）的支持。V1 API 没有被删除，但可能会在这个时间限制外被弃用。
- 所有 **beta** API 通常在九个月或跨三个发行版本（以更长的时间为准）内可用。Beta API 不会在这个时间限制外被删除。
- 所有 **alpha** API 都不是必需的，但如果对用户有好处，则可能会被列为已弃用或删除。

1.1.4.1.1. API 弃用

产品或类别	受影响的项	Version	推荐的操作	详情和链接

1.1.4.1.2. API 删除

产品或类别	受影响的项	Version	推荐的操作	详情和链接

1.1.4.2. 多集群引擎 Operator 弃用

弃用 (deprecated) 组件、功能或服务会被支持，但不推荐使用，并可能在以后的版本中被删除。考虑使用 *推荐操作* 中的相应的替代操作，详情在下表中提供：

产品或类别	受影响的项	Version	推荐的操作	详情和链接

1.1.4.3. 删除

一个 *删除 (removed)* 的项通常是在之前的版本中被弃用的功能，在该产品中不再可用。您必须将 alternatives 用于删除的功能。考虑使用 *推荐操作* 中的相应的替代操作，详情在下表中提供：

产品或类别	受影响的项	Version	推荐的操作	详情和链接
-------	-------	---------	-------	-------

1.2. 关于多集群引擎 OPERATOR 的集群生命周期

multicluster engine for Kubernetes operator 是集群生命周期 Operator，它为 Red Hat OpenShift Container Platform 和 Red Hat Advanced Cluster Management hub 集群提供集群管理功能。如果安装了 Red Hat Advanced Cluster Management，则不需要安装 multicluster engine operator，因为它会被自动安装。

有关支持信息以及以下文档，请参阅 [Kubernetes operator 2.2 的多集群引擎支持列表](#)：

- [要求和建议](#)
- [控制台概述](#)
- [用于 Kubernetes operator 基于角色的访问控制的多集群引擎](#)

要继续，请参阅 [与多集群引擎 operator 的集群生命周期中剩余的集群生命周期文档](#)。

1.2.1. 要求和建议

在安装 multicluster engine operator 前，请查看以下系统配置要求和设置：

- [支持的浏览器和平台](#)
- [网络配置](#)

1.2.1.1. 支持的浏览器和平台

请参阅 [multicluster engine for Kubernetes operator 2.2 支持的浏览器和功能的重要信息](#)。

1.2.1.2. 网络配置

重要：可信 CA 捆绑包在 multicluster engine operator 命名空间中可用，但该增强需要更改您的网络。可信 CA 捆绑包 ConfigMap 使用 **trusted-ca-bundle** 的默认名称。您可以通过在名为 **TRUSTED_CA_BUNDLE** 的环境变量中提供 Operator 来更改此名称。如需更多信息，请参阅 Red Hat OpenShift Container Platform 的 [网络部分](#) 中的 [配置集群范围代理](#)。

注：在受管集群中的注册代理和工作代理不支持代理设置，因为它们通过建立 mTLS 连接与 hub 集群上的 **apiserver** 通信，该连接无法通过代理进行。

将您的网络设置配置为允许以下的连接：

1.2.1.2.1. multicluster engine operator 网络要求

有关 multicluster engine operator 集群网络要求，请查看下表：

方向	连接	端口（如果指定）
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

1.2.2. 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。

1.2.3. multicluster engine operator 基于角色的访问控制

RBAC 在控制台和 API 一级进行验证。控制台中的操作可根据用户访问角色权限启用或禁用。查看以下部分以了解有关产品中特定生命周期的 RBAC 的更多信息：

- [角色概述](#)
- [集群生命周期 RBAC](#)
 - [集群池 RBAC](#)
 - [集群生命周期的控制台和 API RBAC 表](#)
 - [基于角色的凭证访问控制](#)

1.2.3.1. 角色概述

有些产品资源是基于集群范围的，有些则是命名空间范围。您必须将集群角色绑定和命名空间角色绑定应用到用户，以使访问控制具有一致性。查看支持的以下角色定义表列表：

1.2.3.1.1. 角色定义表

角色	定义
cluster-admin	这是 OpenShift Container Platform 的默认角色。具有集群范围内的绑定到 cluster-admin 角色的用户，是一个 OpenShift Container Platform 超级用户，其具有所有访问权限。
open-cluster-management:cluster-manager-admin	具有集群范围内的绑定到 open-cluster-management:cluster-manager-admin 角色的用户，是一个超级用户，其具有所有访问权限。此角色允许用户创建 ManagedCluster 资源。
open-cluster-management:admin:<managed_cluster_name>	具有集群范围内的绑定到 open-cluster-management:admin:<managed_cluster_name> 角色的用户，具有对名为 <managed_cluster_name> 的 ManagedCluster 资源的管理员访问权限。当用户具有受管集群时，会自动创建此角色。
open-cluster-management:view:<managed_cluster_name>	具有集群范围内的绑定到 open-cluster-management:view:<managed_cluster_name> 角色的用户，可以访问名为 <managed_cluster_name> 的 ManagedCluster 资源。
open-cluster-management:managedclusterset:admin:<managed_clusterset_name>	具有集群范围内的绑定到 open-cluster-management:managedclusterset:admin:<managed_clusterset_name> 角色的用户，具有对名为 <managed_clusterset_name> 的 ManagedCluster 资源的管理员访问权限。用户还有对 managedcluster.cluster.open-cluster-management.io 、 clusterclaim.hive.openshift.io 、 clusterdeployment.hive.openshift.io 和 clusterpool.hive.openshift.io 资源的访问权限，这些资源具有受管集群集标签： cluster.open-cluster-management.io 和 clusterset=<managed_clusterset_name> 。使用集群集时会自动生成角色绑定。请参阅 创建 ManagedClusterSet 以了解如何管理该资源。

角色	定义
open-cluster-management:managedclusterset:view:<managed_clusterset_name>	具有集群范围内的绑定到 open-cluster-management:managedclusterset:view:<managed_clusterset_name> 角色的用户，可以访问名为 '<managed_clusterset_name>' 的 ManagedCluster 资源。用户还有对 managedcluster.cluster.open-cluster-management.io 、 clusterclaim.hive.openshift.io 、 clusterdeployment.hive.openshift.io 和 clusterpool.hive.openshift.io 资源的查看访问权限，这些资源具有受管集群集标签： cluster.open-cluster-management.io,clusterset=<managed_clusterset_name> 。有关如何管理受管集群设置资源的更多详细信息，请参阅 创建 ManagedClusterSet 。
admin, edit, view	admin 、 edit 和 view 是 OpenShift Container Platform 的默认角色。具有命名空间范围绑定的用户可以访问特定命名空间中的 open-cluster-management 资源，而集群范围的绑定到同一角色可以访问整个集群范围的 open-cluster-management 资源。

重要：

- 任何用户都可以从 OpenShift Container Platform 创建项目，这为命名空间授予管理员角色权限。
- 如果用户无法访问集群的角色，则无法看到集群名称。集群名称显示有以下符号：-。

RBAC 在控制台和 API 一级进行验证。控制台中的操作可根据用户访问角色权限启用或禁用。查看以下部分以了解有关产品中特定生命周期的 RBAC 的更多信息。

1.2.3.2. 集群生命周期 RBAC

查看以下集群生命周期 RBAC 操作：

- 为所有受管集群创建和管理集群角色绑定。例如，输入以下命令创建到集群角色 **open-cluster-management:cluster-manager-admin** 的集群角色绑定：

```
oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:cluster-manager-admin --user=<username>
```

这个角色是一个超级用户，可访问所有资源和操作。您可以创建集群范围的 **managedcluster** 资源、用于管理受管集群的资源的命名空间，以及使用此角色的命名空间中的资源。您可能需要添加需要角色关联的 ID 用户名，以避免权限错误。

- 运行以下命令，为名为 **cluster-name** 的受管集群管理集群角色绑定：

```
oc create clusterrolebinding (role-binding-name) --clusterrole=open-cluster-management:admin:<cluster-name> --user=<username>
```

此角色对集群范围的 **managedcluster** 资源具有读写访问权限。这是必要的，因为 **managedcluster** 是一个集群范围的资源，而不是命名空间范围的资源。

- 输入以下命令，创建到集群角色 **admin** 的命名空间角色绑定：

```
oc create rolebinding <role-binding-name> -n <cluster-name> --clusterrole=admin --user=<username>
```

此角色对受管集群命名空间中的资源具有读写访问权限。

- 为 **open-cluster-management:view:<cluster-name>** 集群角色创建一个集群角色绑定，以查看名为 **cluster-name** 的受管集群，输入以下命令：

```
oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:view:<cluster-name> --user=<username>
```

此角色具有对集群范围的 **managedcluster** 资源的读取访问权限。这是必要的，因为 **managedcluster** 是一个集群范围的资源。

- 输入以下命令，创建到集群角色 **view** 的命名空间角色绑定：

```
oc create rolebinding <role-binding-name> -n <cluster-name> --clusterrole=view --user=<username>
```

此角色对受管集群命名空间中的资源具有只读访问权限。

- 输入以下命令来查看您可以访问的受管集群列表：

```
oc get managedclusters.clusterview.open-cluster-management.io
```

此命令供没有集群管理员特权的管理员和用户使用。

- 输入以下命令来查看您可以访问的受管集群集列表：

```
oc get managedclustersets.clusterview.open-cluster-management.io
```

此命令供没有集群管理员特权的管理员和用户使用。

1.2.3.2.1. 集群池 RBAC

查看以下集群池 RBAC 操作：

- 作为集群管理员，通过创建受管集群集并使用集群池置备集群，并通过向组添加角色来授予管理员权限。请参见以下示例：

- 使用以下命令为 **server-foundation-clusterset** 受管集群集授予 **admin** 权限：

```
oc adm policy add-cluster-role-to-group open-cluster-management:clusterset-admin:server-foundation-clusterset server-foundation-team-admin
```

- 使用以下命令为 **server-foundation-clusterset** 受管集群授予 **view** 权限：

```
oc adm policy add-cluster-role-to-group open-cluster-management:clusterset-view:server-foundation-clusterset server-foundation-team-user
```

-
- 为集群池 **server-foundation-clusterpool** 创建命名空间。查看以下示例以授予角色权限：
 - 运行以下命令，为 **server-foundation-team-admin** 授予 **server-foundation-clusterpool** 的 **admin** 权限：


```
oc adm new-project server-foundation-clusterpool

oc adm policy add-role-to-group admin server-foundation-team-admin --namespace
server-foundation-clusterpool
```
- 作为团队管理员，在集群池命名空间中创建一个名为 **ocp46-aws-clusterpool** 的集群池，带有集群设置标签 **cluster.open-cluster-management.io/clusterset=server-foundation-clusterset**：
 - **server-foundation-webhook** 检查集群池是否有集群设置标签，以及用户是否有权在集群集中创建集群池。
 - **server-foundation-controller** 为 **server-foundation-team-user** 授予对 **server-foundation-clusterpool** 命名空间的 **view** 权限。
- 创建集群池时，集群池会创建一个 **clusterdeployment**。继续阅读以获取更多详细信息：
 - **server-foundation-controller** 为 **server-foundation-team-admin** 授予对 **clusterdeployment** 命名空间的 **admin** 权限。
 - **server-foundation-controller** 为 **server-foundation-team-user** 授予对 **clusterdeployment** 命名空间的 **view** 权限。
注：作为 **team-admin** 和 **team-user**，您有 **clusterpool**、**clusterdeployment** 和 **clusterclaim** 的 **admin** 权限

1.2.3.2.2. 集群生命周期的控制台和 API RBAC 表

查看以下集群生命周期控制台和 API RBAC 表：

表 1.1. 集群生命周期的控制台 RBAC 表

资源	Admin	Edit	View
Clusters	read、update、delete	-	读取
集群集	get、update、bind、join	未提及 edit 角色	get
受管集群	read、update、delete	未提及 edit 角色	get
AWS 供应商连接。	create、read、update 和 delete	-	读取

表 1.2. 集群生命周期的 API RBAC 表

API	Admin	Edit	View
managedclusters.cluster.open-cluster-management.io 对于这个 API 您可以使用 mcl (单数形式) 或 mcls (复数形式)。	创建、读取、更新、删除	读取、更新	读取
managedclusters.view.open-cluster-management.io 对于这个 API 您可以使用 mcv (单数形式) 或 mcvs (复数形式)。	读取	读取	读取
managedclusters.register.open-cluster-management.io/accept	update	update	
managedclusterset.cluster.open-cluster-management.io 对于这个 API 您可以使用 mclset (单数形式) 或 mclsets (复数形式)。	创建、读取、更新、删除	读取、更新	读取
managedclustersets.view.open-cluster-management.io	读取	读取	读取
managedclustersetbinding.cluster.open-cluster-management.io 对于这个 API 您可以使用 mclsetbinding (单数形式) 或 mclsetbindings (复数形式)。	创建、读取、更新、删除	读取、更新	读取
klusterletaddons.configs.agent.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取

API	Admin	Edit	View
managedclusteractions.action.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取
managedclusterviews.view.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取
managedclusterinfos.internal.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取
manifestworks.work.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取
submarinerconfigs.submarineraddon.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取
placements.cluster.open-cluster-management.io	创建、读取、更新、删除	读取、更新	读取

1.2.3.2.3. 基于角色的凭证访问控制

对凭证的访问由 Kubernetes 控制。凭据作为 Kubernetes secret 存储和保护。以下权限适用于在 Red Hat Advanced Cluster Management for Kubernetes 中访问 secret：

- 有权在命名空间中创建 secret 的用户可以创建凭证。
- 有权读取命名空间中的 secret 的用户也可以查看凭证。
- 具有 Kubernetes 集群角色 **admin** 和 **edit** 的用户可以创建和编辑 secret。
- 具有 Kubernetes 集群角色 **view** 的用户无法查看 secret，因为读取 secret 的内容可以访问服务帐户凭证。

1.3. 安装和升级多集群引擎 OPERATOR

multicluster engine operator 是一个软件 operator，用于增强集群团队管理。multicluster engine operator 支持跨云和数据中心的 Red Hat OpenShift Container Platform 和 Kubernetes 集群生命周期管理。

请参阅以下文档：

- [在线安装](#)
- [在断开连接的网络上安装](#)
- [升级集群](#)
- [卸装](#)
- [MultiClusterEngine 高级配置](#)

1.3.1. 在线安装

multicluster engine operator 安装有 Operator Lifecycle Manager，用于管理安装、升级和删除包含 multicluster engine operator 的组件。

弃用：不再支持 multicluster engine operator 的 2.3 和更早的版本。文档可能仍然可用，但没有任何可用的勘误或其他更新。

最佳实践：升级到最新版本。

需要的访问权限：集群管理员

重要：

- 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，多集群引擎 Operator 组件安装在 OpenShift Container Platform 集群的 worker 节点上，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 将多集群引擎 Operator 安装到 worker 节点上。
- 如果您使用基础架构节点配置了 OpenShift Container Platform 集群，您可以使用带有其他资源参数的 OpenShift Container Platform CLI 将多集群引擎 Operator 安装到这些基础架构节点上。不是所有多集群引擎 Operator 组件都支持基础架构节点，因此在基础架构节点上安装 multicluster engine operator 时仍需要一些 worker 节点。详情请参阅 [在基础架构节点上安装多集群引擎部分](#)。
- 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[高级配置部分](#)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多集群引擎](#)

1.3.1.1. 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 控制台访问 OperatorHub 目录中的 multicluster engine operator。
- 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 4.8](#)
 - [OpenShift Container Platform 版本 4.9](#)
 - [OpenShift Container Platform 版本 4.10](#)
 - [OpenShift Container Platform 版本 4.11](#)
- 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。
- 如果您计划使用 Red Hat OpenShift Container Platform 提供的 Assisted Installer 创建受管集群，请参阅 OpenShift Container Platform 文档中的[使用 Assisted Installer 主题准备安装](#)。

1.3.1.2. 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 multicluster engine operator 尚未安装在 OpenShift Container Platform 集群中。multicluster engine operator 只允许在每个 OpenShift Container Platform 集群中有一个安装。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 console
```

请参见以下示例输出：

```
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>。

您可以继续安装 multicluster engine operator。

1.3.1.3. 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：
 - 命名空间：
 - 多集群引擎 operator 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: {}
```

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

1.3.1.4. 通过 OpenShift Container Platform CLI 安装

- 创建一个 multicluster engine operator 引擎命名空间，其中包含 Operator 的要求。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>
```

- 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>
```

- 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>
  namespace: <namespace>
spec:
  targetNamespaces:
  - <namespace>
```

- 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
```

- 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-2.1
  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。

- 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多集群引擎 Operator，请参阅 [MultiClusterEngine 自定义资源附加配置](#) 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}'
```

如果您要重新安装多集群引擎 operator 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 multicluster engine operator 以及有权访问安装 multicluster engine operator 的命名空间的用户凭证授予集群管理员特权。

1.3.1.5. 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 Subscription 和 **MultiClusterEngine** 自定义资源中。

1.3.1.5.1. 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。

要与多集群引擎 operator 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

```

1.3.1.5.2. 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

```

1.3.1.5.3. 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: ""

```

1.3.2. 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 的 Red Hat Advanced Cluster Management for Kubernetes 的集群上安装多集群引擎 Operator。multicluster engine operator 无法在 2.5 之前的版本上与 Red Hat Advanced Cluster Management for Kubernetes 共存，因为它们提供了一些相同的管理组件。建议您在之前没有安装 Red Hat Advanced Cluster Management 的集群上安装 multicluster engine Operator。如果您在版本 2.5.0 或更高版本中使用 Red Hat Advanced Cluster Management for Kubernetes，则 multicluster engine operator 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](#)
- [在断开连接的环境中安装](#)

1.3.2.1. 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。

1.3.2.2. 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，您可以使用以下命令访问 OpenShift Container Platform Web 控制台来确保正确设置 OpenShift Container Platform 集群：

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

请参见以下示例输出：

```
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。

1.3.2.3. 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安装多集群引擎 Operator：

1. 创建镜像 registry。如果您还没有镜像 registry，请按照 Red Hat OpenShift Container Platform 文档的 [Disconnected 安装镜像](#) 主题中的步骤来创建。
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，以便多集群引擎 operator 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：查找 **registry.redhat.io/quay** 源中的镜像。
- Compliance Operator 策略：查找 **registry.redhat.io/compliance** 源中的镜像。
- Gatekeeper Operator 策略：查找 **registry.redhat.io/mvapich2** 源中的镜像。
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 **mce-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。multicluster engine operator 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。

6. 现在，您在断开连接的 Operator Lifecycle Manager Red Hat Operator 目录中有镜像。您可以使用以下命令来验证：

9. 现在，您在断开连接的 Operator Lifecycle Manager 中已有镜像，从 Operator Lifecycle Manager 目录继续为 Kubernetes 安装 multicluster engine operator。

如需了解所需步骤，[请参阅在线安装](#)。

1.3.3. 升级集群

创建要使用 multicluster engine operator 管理的 Red Hat OpenShift Container Platform 集群后，您可以使用 multicluster engine operator 控制台将这些集群升级到受管集群使用的版本频道中可用的最新次版本。

在连接的环境中，会自动识别更新，并带有为在控制台中需要升级的每个集群提供的通知。

备注：

要升级到一个主要版本，您必须确定是否满足升级到该版本的所有先决条件。在可以使用控制台升级集群前，您必须更新受管集群上的版本频道。

更新受管集群上的版本频道后，多集群引擎 operator 控制台会显示可用于升级的最新版本。

此升级方法只适用于处于 *Ready* 状态的 OpenShift Container Platform 受管集群。

重要：您无法使用 multicluster engine operator 控制台在 Red Hat OpenShift Dedicated 上升级 Red Hat OpenShift Kubernetes Service 受管集群或 OpenShift Container Platform 受管集群。

要在连接的环境中升级集群，请完成以下步骤：

1. 通过导航菜单进入 **Infrastructure > Clusters**。如果有可用的升级，会在 *Distribution version* 列中显示。
2. 选择您要升级的 *Ready* 状态的集群。集群必须是 OpenShift Container Platform 集群才能使用控制台升级。
3. 选择 **Upgrade**。
4. 选择每个集群的新版本。
5. 选择 **Upgrade**。

如果集群升级失败，Operator 通常会重试升级，停止并报告故障组件的状态。在某些情况下，升级过程会一直通过尝试完成此过程进行循环。不支持在失败的升级后将集群还原到以前的版本。如果您的集群升级失败，请联系红帽支持以寻求帮助。

1.3.3.1. 选择一个频道

您可以使用控制台为 OpenShift Container Platform 版本 4.6 或更高版本上的集群升级选择频道。选择频道后，会自动提醒两个勘误版本可用的集群升级（4.8.1 > 4.8.2 > 4.8.3 等）和发行版本（4.8 > 4.9 等）。

要为集群选择频道，请完成以下步骤：

1. 在导航中，选择 **Infrastructure > Clusters**。
2. 选择要更改的集群名称来查看 *Cluster details* 页面。如果集群有一个不同的频道，则 *Channel* 字段中会显示一个编辑图标。
3. 点编辑图标，以修改字段中的设置。

4. 在 *New channel* 字段中选择一个频道。

您可以在集群的 *Cluster details* 页中找到有关可用频道更新的提示信息。

1.3.3.2. 升级断开连接的集群

您可以使用带有多集群引擎 operator 的 Red Hat OpenShift Update Service 来在断开连接的环境中升级集群。

在某些情况下，安全性考虑会阻止集群直接连接到互联网。这使得您很难知道什么时候可以使用升级，以及如何处理这些升级。配置 OpenShift Update Service 可能会有所帮助。

OpenShift Update Service 是一个独立的操作对象，它监控受管集群在断开连接的环境中的可用版本，并使其可用于在断开连接的环境中升级集群。配置 OpenShift Update Service 后，它可以执行以下操作：

- 监测何时适用于断开连接的集群的升级。
- 使用图形数据文件识别哪些更新需要被镜像到您的本地站点进行升级。
- 使用控制台，通知您的集群有可用的升级。

以下主题解释了升级断开连接的集群的步骤：

- [先决条件](#)
- [准备断开连接的镜像 registry](#)
- [为 OpenShift Update Service 部署 Operator](#)
- [构建图形数据 init 容器](#)
- [为已镜像的 registry 配置证书](#)
- [部署 OpenShift Update Service 实例](#)
- [覆盖默认 registry（可选）](#)
- [部署断开连接的目录源](#)
- [更改受管集群参数](#)
- [查看可用升级](#)
- [选择一个频道](#)
- [升级集群](#)

1.3.3.2.1. 先决条件

您必须满足以下先决条件，才能使用 OpenShift Update Service 升级断开连接的集群：

- 部署在 Red Hat OpenShift Container Platform 版本 4.6 或更高版本上运行的 hub 集群，并配置了受限 OLM。如需了解如何 [配置受限 OLM](#) 的详细信息，请参阅 [在受限网络中使用 Operator Lifecycle Manager](#)。
注：在配置受限 OLM 时记录目录源镜像。
- 由 hub 集群管理的 OpenShift Container Platform 集群

- 访问您可以镜像集群镜像的本地存储库的凭证。如需有关如何创建此存储库的更多信息，请参阅 [断开连接的安装镜像](#)。
注：您升级的集群当前版本的镜像必须始终作为镜像的一个镜像可用。如果升级失败，集群会在试图升级时恢复到集群的版本。

1.3.3.2.2. 准备断开连接的镜像 registry

您必须镜像要升级到的镜像，以及您要从本地镜像 registry 升级到的当前镜像。完成以下步骤以镜像镜像：

1. 创建一个包含类似以下示例内容的脚本文件：

```
UPSTREAM_REGISTRY=quay.io
PRODUCT_REPO=openshift-release-dev
RELEASE_NAME=ocp-release
OCP_RELEASE=4.12.2-x86_64
LOCAL_REGISTRY=$(hostname):5000
LOCAL_SECRET_JSON=/path/to/pull/secret ❶

oc adm -a ${LOCAL_SECRET_JSON} release mirror \
--
from=${UPSTREAM_REGISTRY}/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE} \
--to=${LOCAL_REGISTRY}/ocp4 \
--to-release-image=${LOCAL_REGISTRY}/ocp4/release:${OCP_RELEASE}
```

- ❶ 将 `/path/to/pull/secret` 替换为 OpenShift Container Platform pull secret 的路径。

2. 运行该脚本来对镜像进行镜像、配置设置并将发行镜像与发行内容分开。
在创建 `ImageContentSourcePolicy` 时，您可以使用此脚本的最后一行输出。

1.3.3.2.3. 为 OpenShift Update Service 部署 Operator

要在 OpenShift Container Platform 环境中为 OpenShift Update Service 部署 Operator，请完成以下步骤：

1. 在 hub 集群中，访问 OpenShift Container Platform operator hub。
2. 选择 **Red Hat OpenShift Update Service Operator** 来部署 Operator。如果需要，更新默认值。Operator 的部署会创建一个名为 `openshift-cincinnati` 的新项目。
3. 等待 Operator 的安装完成。
您可以通过在 OpenShift Container Platform 命令行中输入 `oc get pods` 命令来检查安装的状态。验证 Operator 是否处于 **running** 状态。

1.3.3.2.4. 构建图形数据 init 容器

OpenShift Update Service 使用图形数据信息来决定可用的升级。在连接的环境中，OpenShift Update Service 会直接从 [Cincinnati 图形数据 GitHub 仓库](#) 中提取可用于升级的图形数据信息。由于要配置断开连接的环境，所以必须使用 `init` 容器使图形数据在本地存储库中可用。完成以下步骤以创建图形数据 `init` 容器：

1. 输入以下命令克隆 `graph data` Git 存储库：

```
git clone https://github.com/openshift/cincinnati-graph-data
```

2. 创建一个包含您的图形数据 **init** 信息的文件。您可以在 **cincinnati-operator GitHub** 仓库中找到此 [Dockerfile](#) 示例。该文件的内容在以下示例中显示：

```
FROM registry.access.redhat.com/ubi8/ubi:8.1 1

RUN curl -L -o cincinnati-graph-data.tar.gz https://github.com/openshift/cincinnati-graph-data/archive/master.tar.gz 2

RUN mkdir -p /var/lib/cincinnati/graph-data/ 3

CMD exec /bin/bash -c "tar xvzf cincinnati-graph-data.tar.gz -C /var/lib/cincinnati/graph-data/ --strip-components=1" 4
```

在本例中：

- 1 **FROM** 值是 OpenShift Update Service 查找镜像的外部 registry。
- 2 **RUN** 命令创建目录并打包升级文件。
- 3 **RUN** 命令创建目录并打包升级文件。
- 4 **CMD** 命令将软件包文件复制到本地库并提取文件进行升级。

3. 运行以下命令来构建 图形数据 **init** 容器：

```
podman build -f <path_to_Dockerfile> -t
<${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container>:latest 1 2
podman push <${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container>
<2>:latest --authfile=</path/to/pull_secret>.json 3
```

- 1 将 **path_to_Dockerfile** 替换为您在上一步中创建文件的路径。
- 2 将 **/\${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container** 替换为本地图形数据 **init** 容器的路径。
- 3 将 **/path/to/pull_secret** 替换为 pull secret 文件的路径。

注：如果没有安装 **podman**，您也可以将命令中的 **podman** 替换为 **docker**。

1.3.3.2.5. 为已镜像的 registry 配置证书

如果您使用安全的外部容器 registry 来存储已镜像的 OpenShift Container Platform 发行镜像，OpenShift Update Service 需要访问此 registry 来构建升级图。完成以下步骤以配置您的 CA 证书以用于 OpenShift Update Service Pod：

1. 查找位于 **image.config.openshift.io** 的 OpenShift Container Platform 外部 registry API。这是存储外部 registry CA 证书的位置。
如需更多信息，请参阅 OpenShift Container Platform 文档中的 [为镜像 registry 访问](#) 配置额外的信任存储。
2. 在 **openshift-config** 命名空间中创建 ConfigMap。

- 在密钥 **updateservice-registry** 中添加您的 CA 证书。OpenShift Update Service 使用此设置来定位您的证书：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: trusted-ca
data:
  updateservice-registry: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

- 编辑 **image.config.openshift.io** API 中的 **cluster** 资源，将 **additionalTrustedCA** 字段设置为您创建的 ConfigMap 的名称。

```
oc patch image.config.openshift.io cluster -p '{"spec":{"additionalTrustedCA":
{"name":"trusted-ca"}}}' --type merge
```

将 **trusted-ca** 替换为新 ConfigMap 的路径。OpenShift Update Service Operator 会监视 **image.config.openshift.io** API 和您在 **openshift-config** 命名空间中创建的 ConfigMap 以获取更改，然后在 CA 证书已更改时重启部署。

1.3.3.2.6. 部署 OpenShift Update Service 实例

当在 hub 集群上完成 OpenShift Update Service 实例部署时，此实例就位于集群升级的镜像镜像位置，并可供断开连接的受管集群使用。完成以下步骤以部署实例：

- 如果您不想使用 Operator 的默认命名空间 (**openshift-cincinnati**)，为 OpenShift Update Service 实例创建一个命名空间：
 - 在 OpenShift Container Platform hub 集群控制台导航菜单中，选择 **Administration > Namespaces**。
 - 点 **Create Namespace**。
 - 添加命名空间的名称以及您的命名空间的任何其他信息。
 - 选择 **Create** 来创建命名空间。
- 在 OpenShift Container Platform 控制台的 *Installed Operators* 部分中，选择 **Red Hat OpenShift Update Service Operator**。
- 在菜单中选择 **Create Instance**。
- 粘贴 OpenShift Update Service 实例中的内容。您的 YAML 实例可能类似以下清单：

```
apiVersion: cincinnati.openshift.io/v1beta2
kind: Cincinnati
metadata:
  name: openshift-update-service-instance
  namespace: openshift-cincinnati
spec:
  registry: <registry_host_name>:<port> 1
```

```
replicas: 1
repository: ${LOCAL_REGISTRY}/ocp4/release
graphDataImage: '<host_name>:<port>/cincinnati-graph-data-container' 2
```

- 1 将 **spec.registry** 值替换为镜像的本地断开连接 registry 的路径。
- 2 将 **spec.graphDataImage** 值替换为图形数据 init 容器的路径。这与运行 **podman push** 命令来推送图形数据 init 容器时使用的值相同。

5. 选择 **Create** 来创建实例。

6. 在 hub 集群 CLI 中输入 **oc get pods** 命令来查看实例创建的状态。它可能需要一段时间，但当命令结果显示实例和运算符正在运行时，进程就会完成。

1.3.3.2.7. 覆盖默认 registry（可选）

注：本节中的步骤只在将发行版本镜像到您的镜像 registry 时才应用。

OpenShift Container Platform 具有一个默认的镜像 registry 值，用于指定它找到升级软件包的位置。在断开连接的环境中，您可以创建一个覆盖，将该值替换为镜像发行镜像的本地镜像 registry 的路径。

完成以下步骤以覆盖默认 registry：

1. 创建名为 **mirror.yaml** 的 YAML 文件，该文件类似于以下内容：

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: <your-local-mirror-name> 1
spec:
  repositoryDigestMirrors:
  - mirrors:
    - <your-registry> 2
    source: registry.redhat.io
```

- 1 将 **your-local-mirror-name** 替换为您的本地镜像的名称。
- 2 将 **your-registry** 替换为本地镜像存储库的路径。

注：您可以通过输入 **oc adm release mirror** 命令来查找到本地镜像的路径。

2. 使用受管集群的命令行，运行以下命令覆盖默认 registry：

```
oc apply -f mirror.yaml
```

1.3.3.2.8. 部署断开连接的目录源

在受管集群中，禁用所有默认目录源并创建一个新源。完成以下步骤，将默认位置从连接的位置改为断开连接的本地 registry：

1. 创建名为 **source.yaml** 的 YAML 文件，该文件类似于以下内容：

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

```

kind: OperatorHub
metadata:
  name: cluster
spec:
  disableAllDefaultSources: true
---
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: '<registry_host_name>:<port>/olm/redhat-operators:v1' ❶
  displayName: My Operator Catalog
  publisher: grpc

```

❶ 将 **spec.image** 值替换为本地受限目录源镜像的路径。

2. 在受管集群的命令行中，运行以下命令来更改目录源：

```
oc apply -f source.yaml
```

1.3.3.2.9. 更改受管集群参数

更新受管集群的 **ClusterVersion** 资源信息，以更改从中检索其升级的默认位置。

1. 在受管集群中，输入以下命令确认 **ClusterVersion** 上游参数目前是默认的公共 OpenShift Update Service 操作对象：

```
oc get clusterversion -o yaml
```

返回的内容可能类似以下内容：

```

apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: ClusterVersion
  [..]
  spec:
    channel: stable-4.12
    upstream: https://api.openshift.com/api/upgrades_info/v1/graph

```

2. 在 hub 集群中，输入以下命令识别 OpenShift Update Service 操作对象的路由 URL：

```
oc get routes
```

记录返回的值以获取后续步骤。

3. 在受管集群的命令行中，输入以下命令编辑 **ClusterVersion** 资源：

```
oc edit clusterversion version
```

将 `spec.channel` 的值替换为您的新版本。

将 `spec.upstream` 值替换为 hub 集群 OpenShift Update Service 操作对象的路径。您可以完成以下步骤来确定操作对象的路径：

- a. 在 hub 集群中运行以下命令：

```
oc get routes -A
```

- b. 查找到 `cincinnati` 的路径。操作对象的路径是 `HOST/PORT` 字段中的值。

4. 在受管集群的命令行中，输入以下命令确认 `ClusterVersion` 中的 `upstream` 参数已更新为本地 hub 集群 OpenShift Update Service URL：

```
oc get clusterversion -o yaml
```

结果类似以下内容：

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: ClusterVersion
  [...]
  spec:
    channel: stable-4.12
    upstream: https://<hub-cincinnati-uri>/api/upgrades_info/v1/graph
```

1.3.3.2.10. 查看可用升级

在 *Clusters* 页面中，如果断开连接的 registry 中有升级，集群的 **Distribution version** 表示有一个可用的升级。您可以通过从 *Actions* 菜单中选择集群并选择 **Upgrade cluster** 来查看可用的升级。如果可选的升级路径可用，则会列出可用的升级。

注：如果当前版本没有镜像到本地镜像存储库，则不会显示可用的升级版本。

1.3.3.2.11. 选择一个频道

您可以使用控制台为 OpenShift Container Platform 版本 4.6 或更高版本上的集群升级选择频道。这些版本必须在镜像 registry 上可用。完成 [选择频道](#) 中的步骤为您的升级指定频道。

1.3.3.2.12. 升级集群

配置断开连接的 registry 后，多集群引擎 operator 和 OpenShift Update Service 使用断开连接的 registry 来确定升级是否可用。如果没有可用的升级，请确保您有集群当前级别的发行镜像，且至少有一个后续级别的镜像位于本地存储库中。如果集群当前版本的发行镜像不可用，则没有可用的升级。

在 *Clusters* 页面中，如果断开连接的 registry 中有升级，集群的 **Distribution version** 表示有一个可用的升级。您可以通过点 **Upgrade available** 并选择升级的版本来升级镜像。受管集群已更新至所选版本。

如果集群升级失败，Operator 通常会重试升级，停止并报告故障组件的状态。在某些情况下，升级过程会一直通过尝试完成此过程进行循环。不支持在失败的升级后将集群还原到以前的版本。如果您的集群升级失败，请联系红帽支持以寻求帮助。

1.3.4. 高级配置

multicluster engine operator 使用部署所有所需组件的 operator 安装。通过在 **MultiClusterEngine** 自定义资源中添加一个或多个以下属性，可以在安装过程中或安装后进一步配置 multicluster engine operator：

1.3.4.1. local-cluster 启用

默认情况下，运行 multicluster engine operator 的集群管理其自身。要在没有集群管理其自身的情况下安装 multicluster engine operator，在 **MultiClusterEngine** 部分中的 **spec.overrides.components** 设置中指定以下值：

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
      - name: local-cluster
        enabled: false
```

- **name** 值将 hub 集群标识为 **local-cluster**。
- **enabled** 设置指定功能是启用还是禁用。当值为 **true** 时，hub 集群会自己管理。当值为 **false** 时，hub 集群不自己管理。

由自身管理的 hub 集群在集群列表中被指定为 **local-cluster**。

1.3.4.2. 自定义镜像 pull secret

如果您计划导入不是由 OpenShift Container Platform 或 multicluster engine operator 创建的 Kubernetes 集群，请生成一个包含 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: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  imagePullSecret: <secret>

```

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

```

1.3.4.4. availabilityConfig

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

```

1.3.4.5. 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 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: ""

```

1.3.4.6. 容忍 (tolerations)

您可以定义容忍列表，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。

1.3.4.7. 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** 规格中包含以下内容：

```
apiVersion: multicluster.openshift.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}}]'
```

1.3.4.8. 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: multicluster.openshift.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}}]'
```

1.3.5. 卸载

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。

1.3.5.1. 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。

1.3.5.2. 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。

1.3.5.3. 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。

1.3.5.4. 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
```

https://access.redhat.com/documentation/zh-cn/openshift_container_platform/4.11/html/installing/disconnected-installation-mirroring

1.4. 管理凭证

在使用 multicluster engine operator 的云服务供应商上创建和管理 Red Hat OpenShift Container Platform 集群需要一个凭证。凭证存储云提供商的访问信息。每个提供程序帐户都需要自己的凭证，就像单个提供程序中的每个域一样。

您可以创建和管理集群凭证。凭证存储为 Kubernetes secret。secret 复制到受管集群的命名空间，以便受管集群的控制器可以访问 secret。更新凭证时，secret 的副本会在受管集群命名空间中自动更新。

注：对现有受管集群，对 pull secret、SSH 密钥或基域的更改不会反映，因为它们已使用原始凭证置备。

需要的访问权限：Edit

- [为 Amazon Web Services 创建凭证](#)

- 为 Microsoft Azure 创建凭证
- 为 Google Cloud Platform 创建凭证
- 为 VMware vSphere 创建凭证
- 为 Red Hat OpenStack Platform 创建凭证
- 为 Red Hat Virtualization 创建凭证
- 为 Red Hat OpenShift Cluster Manager 创建凭证
- 为 Ansible Automation Platform 创建凭证
- 为内部环境创建凭证

1.4.1. 为 Amazon Web Services 创建凭证

您需要一个凭证才能使用多集群引擎 operator 控制台在 Amazon Web Services (AWS) 上部署和管理 Red Hat OpenShift Container Platform 集群。

需要的访问权限：Edit

注：必须在使用 multicluster engine operator 创建集群前完成此步骤。

1.4.1.1. 先决条件

创建凭证前必须满足以下先决条件：

- 已部署多集群引擎 operator hub 集群
- 可通过互联网访问 multicluster engine operator hub 集群，以便它在 Amazon Web Services (AWS) 上创建 Kubernetes 集群
- AWS 登录凭证，其中包括访问密钥 ID 和 secret 访问密钥。请参阅[了解和获取您的安全凭证](#)。
- 允许在 AWS 上安装集群的帐户权限。有关如何配置的说明，请参阅[配置 AWS 帐户](#)。

1.4.1.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

您可以选择为您的凭证添加基本 DNS 域。如果您将基本 DNS 域添加到凭证中，则在使用此凭证创建集群时，会自动填充到正确的字段中。请参见以下步骤：

1. 为您的 AWS 帐户添加 AWS 访问密钥 ID。登录 [AWS](#) 以查找您的 ID。
2. 提供新 AWS Secret 访问密钥的内容。
3. 如果要启用代理，请输入代理信息：
 - HTTP 代理 URL：用作 HTTP 流量的代理的 URL。

- HTTPS 代理 URL：用于 **HTTPS** 流量的安全代理 URL。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
 - 无代理域：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加星号 (*) 以绕过所有目的地的代理。
 - 其他信任捆绑包：代理 HTTPS 连接所需的一个或多个额外 CA 证书。
4. 输入您的 *Red Hat OpenShift pull secret*。您可以从 [Pull secret](#) 下载 pull secret。
 5. 添加可让您连接到集群的 *SSH 私钥*和 *SSH 公钥*。您可以使用现有密钥对，或使用密钥生成程序创建新密钥对。

如需有关如何 [生成密钥的更多信息](#)，请参阅[生成 SSH 私钥并将其添加到代理中](#)。

要创建使用此凭证的集群，完成 [Creating a cluster on Amazon Web Services](#) 或 [Creating a cluster on Amazon Web Services GovCloud](#) 中的步骤。

您可以在控制台中编辑凭证。如果使用这个供应商连接创建集群，则来自 `<cluster-namespace>` 的 `<cluster-name>-aws-creds` secret 将使用新凭证进行更新。

注：更新凭证不适用于集群池声明的集群。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.1.3. 使用 API 创建不透明的 secret

要使用 API 为 Amazon Web Services 创建不透明 secret，请在类似以下示例的 YAML preview 窗口中应用 YAML 内容：

```
kind: Secret
metadata:
  name: <managed-cluster-name>-aws-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  aws_access_key_id: $(echo -n "${AWS_KEY}" | base64 -w0)
  aws_secret_access_key: $(echo -n "${AWS_SECRET}" | base64 -w0)
```

备注：

- Opaque secret 在控制台中是不可见的。
- 不透明 secret 在您选择的受管集群命名空间中创建。Hive 使用 opaque secret 来置备集群。当使用 Red Hat Advanced Cluster Management 控制台置备集群时，您预先创建的凭证将复制到受管集群命名空间中，作为 opaque secret。

1.4.2. 为 Microsoft Azure 创建凭证

您需要一个凭证来使用多集群引擎 operator 控制台在 Microsoft Azure 或 Microsoft Azure Government 上创建和管理 Red Hat OpenShift Container Platform 集群。

需要的访问权限：Edit

注：这个过程是使用多集群引擎 operator 创建集群的先决条件。

1.4.2.1. 先决条件

创建凭证前必须满足以下先决条件：

- 已部署多集群引擎 operator hub 集群。
- 可通过互联网访问 multicluster engine operator hub 集群，以便它在 Azure 上创建 Kubernetes 集群。
- Azure 登录凭证，其中包括您的基域资源组和 Azure Service Principal JSON。请参阅 azure.microsoft.com。
- 允许在 Azure 上安装集群的帐户权限。如需更多信息，请参阅 [如何配置云服务](#)和[配置 Azure 帐户](#)。

1.4.2.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

1. 可选：为您的凭证添加 **基本 DNS 域**。如果您将基本 DNS 域添加到凭证中，则在使用此凭证创建集群时，会自动填充到正确的字段中。
2. 选择集群的环境是 **AzurePublicCloud** 还是 **AzureUSrianCloud**。Azure Government 环境的设置有所不同，以确保正确设置了此设置。
3. 为您的 Azure 帐户添加 **基本域资源组名称**。此条目是您使用 Azure 帐户创建的资源名称。您可以在 Azure 界面中选择 **Home > DNS Zones** 来查找您的基域资源组名称。请参阅使用 [Azure CLI 创建 Azure 服务主体](#)，以查找您的基域资源组名称。
4. 为您的 **客户端 ID** 提供内容。当您使用以下命令创建服务主体时，这个值作为 **appld** 属性被生成：

```
az ad sp create-for-rbac --role Contributor --name <service_principal> --scopes
<subscription_path>
```

将 *service_principal* 替换为您的服务主体名。

5. 添加您的 **客户端 Secret**。当您使用以下命令创建服务主体时，这个值作为 **password** 属性被生成：

```
az ad sp create-for-rbac --role Contributor --name <service_principal> --scopes
<subscription_path>
```

将 *service_principal* 替换为您的服务主体名。如需了解更多详细信息，请参阅使用 [Azure CLI 创建 Azure 服务主体](#)。

6. 添加您的 **订阅 ID**。这个值是以下命令输出中的 **id** 属性：

```
az account show
```

7. 添加您的 **租户 ID**。这个值是以下命令输出中的 **tenantId** 属性：

```
az account show
```

8. 如果要启用代理，请输入代理信息：

- HTTP 代理 URL：用作 **HTTP** 流量的代理的 URL。
- HTTPS 代理 URL：用于 **HTTPS** 流量的安全代理 URL。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- 无代理域：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加星号 (*) 以绕过所有目的地的代理。
- 其他信任捆绑包：代理 HTTPS 连接所需的一个或多个额外 CA 证书。

9. 输入您的 *Red Hat OpenShift pull secret*。您可以从 [Pull secret](#) 下载 pull secret。

10. 添加用于连接到集群的 *SSH 私钥*和 *SSH 公钥*。您可以使用现有密钥对，或使用密钥生成程序创建新密钥对。如需有关如何 [生成密钥的更多信息](#)，请参阅[生成 SSH 私钥并将其添加到代理中](#)。

要创建使用此凭证的集群，您可以完成在 [Microsoft Azure 上创建集群](#) 中的步骤。

您可以在控制台中编辑凭证。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.2.3. 使用 API 创建不透明的 secret

要使用 API 而不是控制台为 Microsoft Azure 创建不透明 secret，请在类似以下示例的 YAML preview 窗口中应用 YAML 内容：

```
kind: Secret
metadata:
  name: <managed-cluster-name>-azure-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  baseDomainResourceGroupName: $(echo -n "${azure_resource_group_name}" | base64 -w0)
  osServicePrincipal.json: $(base64 -w0 "${AZURE_CRED_JSON}")
```

备注：

- Opaque secret 在控制台中是不可见的。
- 不透明 secret 在您选择的受管集群命名空间中创建。Hive 使用 opaque secret 来置备集群。当使用 Red Hat Advanced Cluster Management 控制台置备集群时，您预先创建的凭证将复制到受管集群命名空间中，作为 opaque secret。

1.4.3. 为 Google Cloud Platform 创建凭证

您需要一个凭证来使用 multicluster engine operator 控制台在 Google Cloud Platform (GCP)上创建和管理 Red Hat OpenShift Container Platform 集群。

需要的访问权限：Edit

注：这个过程是使用多集群引擎 operator 创建集群的先决条件。

1.4.3.1. 先决条件

创建凭证前必须满足以下先决条件：

- 已部署多集群引擎 operator hub 集群
- 可通过互联网访问 multicluster engine operator hub 集群，以便它在 GCP 上创建 Kubernetes 集群
- GCP 登录凭证，其中包括用户 Google Cloud Platform 项目 ID 和 Google Cloud Platform 服务帐户 JSON 密钥。请参阅[创建和管理项目](#)。
- 允许在 GCP 上安装集群的帐户权限。有关如何配置帐户的说明，请参阅[配置 GCP 项目](#)。

1.4.3.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便和安全起见，创建一个命名空间，专门用于托管您的凭证。

您可以选择为您的凭证添加基本 DNS 域。如果您将基本 DNS 域添加到凭证中，则在使用此凭证创建集群时，会自动填充到正确的字段中。请参见以下步骤：

1. 为您的 GCP 帐户添加 Google Cloud Platform 项目 ID。登录到 [GCP](#) 以检索您的设置。
2. 添加 Google Cloud Platform 服务帐户 JSON 密钥。请参阅 (<https://cloud.google.com/iam/docs/creating-managing-service-accounts>) 创建服务帐户 JSON 密钥。按照 GCP 控制台的步骤进行操作。
3. 提供您的新 Google Cloud Platform 服务帐户 JSON 密钥的内容。
4. 如果要启用代理，请输入代理信息：
 - HTTP 代理 URL：用作 HTTP 流量的代理的 URL。
 - HTTPS 代理 URL：用于 HTTPS 流量的安全代理 URL。如果没有提供值，则使用相同的值 HTTP Proxy URL，用于 HTTP 和 HTTPS。
 - 无代理域：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加星号 (*) 以绕过所有目的地的代理。
 - 其他信任捆绑包：代理 HTTPS 连接所需的一个或多个额外 CA 证书。
5. 输入您的 Red Hat OpenShift pull secret。您可以从 [Pull secret](#) 下载 pull secret。
6. 添加 SSH 私钥和 SSH 公钥以便您访问集群。您可以使用现有密钥对，或使用密钥生成程序创建新密钥对。

如需有关如何 [生成密钥的更多信息](#)，请参阅[生成 SSH 私钥并将其添加到代理中](#)。

要在创建集群时使用此连接，您可以完成在 [Google Cloud Platform 上创建集群](#) 中的步骤。

您可以在控制台中编辑凭证。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.3.3. 使用 API 创建不透明的 secret

要使用 API 而不是控制台为 Google Cloud Platform 创建不透明 secret，请在类似以下示例的 YAML preview 窗口中应用 YAML 内容：

```
kind: Secret
metadata:
  name: <managed-cluster-name>-gcp-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  osServiceAccount.json: $(base64 -w0 "${GCP_CRED_JSON}")
```

备注：

- Opaque secret 在控制台中是不可见的。
- 不透明 secret 在您选择的受管集群命名空间中创建。Hive 使用 opaque secret 来置备集群。当使用 Red Hat Advanced Cluster Management 控制台置备集群时，您预先创建的凭证将复制到受管集群命名空间中，作为 opaque secret。

1.4.4. 为 VMware vSphere 创建凭证

您需要一个凭证才能使用多集群引擎 operator 控制台在 VMware vSphere 上部署和管理 Red Hat OpenShift Container Platform 集群。注：仅支持 OpenShift Container Platform 版本 4.5.x 及更新的版本。

需要的访问权限：Edit

注：必须在使用 multicluster engine operator 创建集群前完成此步骤。

1.4.4.1. 先决条件

创建凭证前必须满足以下先决条件：

- 在 OpenShift Container Platform 版本 4.6 或更高版本上部署了 hub 集群。
- hub 集群的互联网访问，以便它在 VMware vSphere 上创建 Kubernetes 集群。
- 使用安装程序置备的基础架构时为 OpenShift Container Platform 配置了 VMware vSphere 登录凭证和 vCenter 要求。请参阅[使用自定义在 vSphere 上安装集群](#)。这些凭证包括以下信息：
 - vCenter 帐户权限。
 - 集群资源。
 - DHCP 可用。
 - ESXi 主机的时间已同步（例如：NTP）。

1.4.4.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

您可以选择为您的凭证添加基本 DNS 域。如果您将基本 DNS 域添加到凭证中，则在使用此凭证创建集群时，会自动填充到正确的字段中。请参见以下步骤：

1. 添加 VMware vCenter 服务器完全限定主机名或 IP 地址。该值必须在 vCenter 服务器 root CA 证书中定义。如果可能，请使用完全限定主机名。
2. 添加 VMware vCenter 用户名。
3. 添加 VMware vCenter 密码。
4. 添加 VMware vCenter root CA 证书。
 - a. 您可以使用 VMware vCenter 服务器的证书在 **download.zip** 软件包中下载证书，地址为 https://<vCenter_address>/certs/download.zip。将 `vCenter_address` 替换为 vCenter 服务器的地址。
 - b. 解包 **download.zip**。
 - c. 使用 **certs/<platform>** 目录中有 **.0** 扩展名的证书。提示：您可以使用 **ls certs/<platform>** 命令列出平台的所有可用证书。
将 **<platform>** 替换为您的平台的缩写：**lin**、**mac** 或 **win**。

例如：**certs/lin/3a343545.0**

最佳实践：使用以下命令将带有 **.0** 扩展的多个证书链接到一起：

```
cat certs/lin/*.0 > ca.crt
```

5. 添加 VMware vSphere 集群名称。
6. 添加 VMware vSphere 数据中心。
7. 添加 VMware vSphere 默认数据存储。
8. 添加 VMware vSphere 磁盘类型。
9. 添加 VMware vSphere 文件夹。
10. 添加 VMware vSphere 资源池。
11. 只用于断开连接的安装：使用所需信息完成 **Configuration for disconnected installation** 子字段：
 - **镜像内容源**：此值包含断开连接的 registry 路径。该路径包含所有用于断开连接的安装镜像的主机名、端口和库路径。示例：**repository.com:5000/openshift/ocp-release**。
该路径会在 **install-config.yaml** 中创建一个到 Red Hat OpenShift Container Platform 发行镜像的镜像内容源策略映射。例如，**repository.com:5000** 生成此 **imageContentSource** 内容：

```
- mirrors:
  - registry.example.com:5000/ocp4
    source: quay.io/openshift-release-dev/ocp-release-nightly
- mirrors:
  - registry.example.com:5000/ocp4
    source: quay.io/openshift-release-dev/ocp-release
```

```
- mirrors:
  - registry.example.com:5000/ocp4
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

- *Additional trust bundle* : 此值提供访问镜像 registry 所需的证书文件内容。
注：如果您要从断开连接的环境中的一个 hub 部署受管集群，并希望在安装后自动导入它们，使用 **YAML** 编辑器将镜像内容源策略添加到 **install-config.yaml** 文件中。以下示例中显示了一个示例：

```
- mirrors:
  - registry.example.com:5000/rhacm2
    source: registry.redhat.io/rhacm2
```

12. 如果要启用代理，请输入代理信息：

- HTTP 代理 URL：用作 **HTTP** 流量的代理的 URL。
- HTTPS 代理 URL：用于 **HTTPS** 流量的安全代理 URL。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- 无代理域：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加星号 (*) 以绕过所有目的地的代理。
- 其他信任捆绑包：代理 HTTPS 连接所需的一个或多个额外 CA 证书。

13. 输入您的 *Red Hat OpenShift pull secret*。您可以从 [Pull secret](#) 下载 pull secret。

14. 添加可让您连接到集群的 *SSH 私钥*和 *SSH 公钥*。

您可以使用现有密钥对，或使用密钥生成程序创建新密钥对。如需更多信息，请参阅[为集群节点 SSH 访问生成密钥对](#)。

要创建使用此凭证的集群，您可以完成在 [VMware vSphere 上创建集群](#) 中的步骤。

您可以在控制台中编辑凭证。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.4.3. 使用 API 创建不透明的 secret

要使用 API 而不是控制台为 VMware vSphere 创建不透明 secret，请在类似以下示例的 YAML preview 窗口中应用 YAML 内容：

```
kind: Secret
metadata:
  name: <managed-cluster-name>-vsphere-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  username: $(echo -n "${VMW_USERNAME}" | base64 -w0)
  password.json: $(base64 -w0 "${VMW_PASSWORD}")
```

备注：

- Opaque secret 在控制台中是不可见的。

- 不透明 secret 在您选择的受管集群命名空间中创建。Hive 使用 opaque secret 来置备集群。当使用 Red Hat Advanced Cluster Management 控制台置备集群时，您预先创建的凭证将复制到受管集群命名空间中，作为 opaque secret。

1.4.5. 为 Red Hat OpenStack 创建凭证

您需要一个凭证才能使用多集群引擎 operator 控制台在 Red Hat OpenStack Platform 上部署和管理 Red Hat OpenShift Container Platform 集群。注：仅支持 OpenShift Container Platform 版本 4.5.x 及更新的版本。

注：必须在使用 multicluster engine operator 创建集群前完成此步骤。

1.4.5.1. 先决条件

创建凭证前必须满足以下先决条件：

- 在 OpenShift Container Platform 版本 4.6 或更高版本上部署了 hub 集群。
- 可通过互联网访问 hub 集群，以便它在 Red Hat OpenStack Platform 上创建 Kubernetes 集群。
- 使用安装程序置备的基础架构时，为 OpenShift Container Platform 配置 Red Hat OpenStack Platform 登录凭证和 Red Hat OpenStack Platform 要求。请参阅[使用自定义配置在 OpenStack 上安装集群](#)。
- 下载或创建 **clouds.yaml** 文件来访问 CloudStack API。在 **clouds.yaml** 文件中：
 - 确定要使用的云身份验证部分名称。
 - 在 **username** 行后马上添加一个 **password** 行。

1.4.5.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

1. 添加 Red Hat OpenStack Platform **cloud.yaml** 文件内容。**clouds.yaml** 文件的内容（包括密码）提供了连接到 Red Hat OpenStack Platform 服务器所需的信息。文件内容必须包含密码，您可以在用户名后马上添加到新行中。
2. 对于使用内部证书颁发机构的配置，请修改 **clouds.yaml** 文件来引用 Hive 部署器 pod 中证书捆绑包的最终位置。Hive 将证书捆绑包 secret 挂载到部署器 pod 中的 **/etc/openstack-ca** 中。该目录中的文件对应于创建集群时提供的 secret 中的键。假设 secret 中使用了密钥 **ca.crt**，将 **cacert** 参数添加到 **clouds.yaml** 文件中，如下例所示：

```
clouds:
  openstack:
    auth:
      auth_url: https://openstack.example.local:13000
      username: "svc-openshift"
      project_id: aa0wet0wfwjerj
      user_domain_name: "idm"
      password: REDACTED
      region_name: "regionOne"
```

```
interface: "public"
identity_api_version: 3
cacert: /etc/openshift-ca/ca.crt
```

3. 添加您的 Red Hat OpenStack Platform 名称。此条目是在 **clouds.yaml** 的 **cloud** 部分中指定的名称，用于建立与 Red Hat OpenStack Platform 服务器的通信。
4. 您可以选择为您的凭证添加基本 DNS 域。如果您将基本 DNS 域添加到凭证中，则在使用此凭证创建集群时，会自动填充到正确的字段中。
5. 只用于断开连接的安装：使用所需信息完成 **Configuration for disconnected installation** 子字段：

- **Cluster OS image**：此值包含用于 Red Hat OpenShift Container Platform 集群机器的镜像的 URL。
- **镜像内容源**：此值包含断开连接的 registry 路径。该路径包含所有用于断开连接的安装镜像的主机名、端口和库路径。示例：**repository.com:5000/openshift/ocp-release**。该路径会在 **install-config.yaml** 中创建一个到 Red Hat OpenShift Container Platform 发行镜像的镜像内容源策略映射。例如，**repository.com:5000** 生成此 **imageContentSource** 内容：

```
- mirrors:
  - registry.example.com:5000/ocp4
  source: quay.io/openshift-release-dev/ocp-release-nightly
- mirrors:
  - registry.example.com:5000/ocp4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - registry.example.com:5000/ocp4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

- **Additional trust bundle**：此值提供访问镜像 registry 所需的证书文件内容。
注：如果您要从断开连接的环境中的一个 hub 部署受管集群，并希望在安装后自动导入它们，使用 **YAML** 编辑器将镜像内容源策略添加到 **install-config.yaml** 文件中。以下示例中显示了一个示例：

```
- mirrors:
  - registry.example.com:5000/rhacm2
  source: registry.redhat.io/rhacm2
```

6. 如果要启用代理，请输入代理信息：
 - **HTTP 代理 URL**：用作 **HTTP** 流量的代理的 URL。
 - **HTTPS 代理 URL**：用于 **HTTPS** 流量的安全代理 URL。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
 - **无代理域**：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加星号 (*) 以绕过所有目的地的代理。
 - **其他信任捆绑包**：代理 **HTTPS** 连接所需的一个或多个额外 CA 证书。
7. 输入 Red Hat OpenShift Pull Secret。您可以从 [Pull secret](#) 下载 pull secret。

8. 添加可让您连接到集群的 SSH 私钥和 SSH 公钥。您可以使用现有密钥对，或使用密钥生成程序创建新密钥对。如需更多信息，请参阅[为集群节点 SSH 访问生成密钥对](#)。
9. 点 **Create**。
10. 查看新凭据信息，然后单击 **Add**。添加凭证时，会将其添加到凭证列表中。

要创建使用此凭证的集群，您可以完成在 [Red Hat OpenStack Platform 上创建集群](#) 中的步骤。

您可以在控制台中编辑凭证。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.5.3. 使用 API 创建不透明的 secret

要使用 API 而不是控制台为 Red Hat OpenStack Platform 创建不透明 secret，请在类似以下示例的 YAML preview 窗口中应用 YAML 内容：

```
kind: Secret
metadata:
  name: <managed-cluster-name>-osp-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  clouds.yaml: $(base64 -w0 "${OSP_CRED_YAML}") cloud: $(echo -n "openstack" | base64 -w0)
```

备注：

- Opaque secret 在控制台中是不可见的。
- 不透明 secret 在您选择的受管集群命名空间中创建。Hive 使用 opaque secret 来置备集群。当使用 Red Hat Advanced Cluster Management 控制台置备集群时，您预先创建的凭证将复制到受管集群命名空间中，作为 opaque secret。

1.4.6. 为 Red Hat Virtualization 创建凭证

您需要一个凭证才能使用多集群引擎 operator 控制台在 Red Hat Virtualization 上部署和管理 Red Hat OpenShift Container Platform 集群。

注：必须在使用 multicluster engine operator 创建集群前完成此步骤。

1.4.6.1. 先决条件

创建凭证前必须满足以下先决条件：

- 在 OpenShift Container Platform 版本 4.7 或更高版本上部署了 hub 集群。
- 可通过互联网访问 hub 集群，以便它在 Red Hat Virtualization 上创建 Kubernetes 集群。
- 配置 Red Hat Virtualization 环境的 Red Hat Virtualization 登录凭证。请参阅 Red Hat Virtualization 文档中的[安装指南](#)。以下列表显示所需信息：
 - oVirt URL
 - oVirt 完全限定域名 (FQDN)

- oVirt 用户名
- oVirt 密码
- oVirt CA/证书
- 可选：代理信息（如果启用了代理）。
- Red Hat OpenShift Container Platform pull secret 信息。您可以从 [Pull secret](#) 下载 pull secret。
- 用于传输最终集群的 SSH 私钥和公钥。
- 允许在 oVirt 上安装集群的帐户权限。

1.4.6.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

1. 为新凭证添加基本信息。当使用此凭证创建集群时，您可以选择添加基本 DNS 域，该域会在正确的字段中自动填充。如果您没有将其添加到凭证中，您可以在创建集群时添加它。
2. 为 Red Hat Virtualization 环境添加所需信息。
3. 如果要启用代理，请输入代理信息：
 - HTTP Proxy URL：应该用作 HTTP 流量的代理的 URL。
 - HTTPS Proxy URL：在使用 HTTPS 流量时应使用的安全代理 URL。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 HTTP 和 HTTPS。
 - 无代理域：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加星号 (*) 以绕过所有目的地的代理。
4. 输入 Red Hat OpenShift Container Platform pull secret。您可以从 [Pull secret](#) 下载 pull secret。
5. 添加可让您连接到集群的 SSH 私钥和 SSH 公钥。您可以使用现有密钥对，或使用密钥生成程序创建新密钥对。如需更多信息，请参阅[为集群节点 SSH 访问生成密钥对](#)。
6. 查看新凭据信息，然后单击 **Add**。添加凭证时，会将其添加到凭证列表中。

要创建使用此凭证的集群，您可以完成在 [Red Hat Virtualization 上创建集群](#) 中的步骤。

您可以在控制台中编辑凭证。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.7. 为 Red Hat OpenShift Cluster Manager 创建凭证

添加 OpenShift Cluster Manager 凭证，以便您可以发现集群。

需要的访问权限: Administrator

1.4.7.1. 先决条件

您需要一个 console.redhat.com 帐户。稍后，您将需要这个值，它可从 console.redhat.com/openshift/token 获取。

1.4.7.2. 使用控制台管理凭证

您需要添加凭证来发现集群。要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

输入 OpenShift Cluster Manager API 令牌，该令牌可以从 console.redhat.com/openshift/token 获取。

您可以在控制台中编辑凭证。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

如果您的凭证被删除，或者 OpenShift Cluster Manager API 令牌已过期或被撤销，则会删除相关的发现集群。

1.4.8. 为 Ansible Automation Platform 创建凭证

您需要一个凭证才能使用多集群引擎 operator 控制台来部署和管理使用 Red Hat Ansible Automation Platform 的 Red Hat OpenShift Container Platform 集群。

需要的访问权限：Edit

注：必须在创建 Automation 模板前完成此步骤，以便在集群中启用自动化。

1.4.8.1. 先决条件

创建凭证前必须满足以下先决条件：

- 已部署多集群引擎 operator hub 集群
- 多集群引擎 operator hub 集群的互联网访问
- Ansible 登录凭证，其中包括 Ansible Automation Platform 主机名和 OAuth 令牌；请参阅 [Ansible Automation Platform 的凭证](#)。
- 允许您安装 hub 集群并使用 Ansible 的帐户权限。了解有关 [Ansible 用户](#) 的更多信息。

1.4.8.2. 使用控制台管理凭证

要从 multicluster engine operator 控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

您在创建 Ansible 凭据时提供的 Ansible 令牌和主机 URL 会自动更新，以便在您编辑凭据时使用该凭据的自动化。更新复制到任何使用 Ansible 凭据的自动化中，包括与集群生命周期、监管和应用程序管理自动化相关的自动化。这可确保自动化在更新凭证后继续运行。

您可以在控制台中编辑凭证。Ansible 凭据在自动化中自动更新，您在凭据中更新该凭据时使用该凭据。

要创建使用此凭证的 Ansible 作业，您可以完成 [配置 Ansible Automation Platform 任务以在受管集群上运行](#) 的步骤。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.4.9. 为内部环境创建凭证

您需要一个凭证才能使用控制台在内部环境中部署和管理 Red Hat OpenShift Container Platform 集群。凭证指定用于集群的连接。

需要的访问权限：Edit

- [先决条件](#)
- [使用控制台管理凭证](#)

1.4.9.1. 先决条件

创建凭证前需要满足以下先决条件：

- 已部署 hub 集群。
- 可通过互联网访问 hub 集群，以便它在基础架构环境中创建 Kubernetes 集群。
- 对于断开连接的环境，您必须配置了一个镜像 registry，您可以在其中复制发行镜像以进行集群创建。如需更多信息，请参阅 [OpenShift Container Platform 文档中的用于断开连接的安装的镜像](#)。
- 支持在内部环境中安装集群的帐户权限。

1.4.9.2. 使用控制台管理凭证

要从控制台创建凭证，请完成控制台中的步骤。

从导航菜单开始。单击 **Credentials** 以从现有凭证选项中进行选择。提示：为方便起见，同时为了提高安全性，创建一个命名空间，专门用于托管您的凭证。

1. 为您的凭证类型选择 **Host inventory**。
2. 您可以选择为您的凭证添加 **基本 DNS 域**。如果您将基本 DNS 域添加到凭证中，则在使用此凭证创建集群时，会自动填充到正确的字段中。如果没有添加 DNS 域，您可以在创建集群时添加它。
3. 输入您的 *Red Hat OpenShift pull secret*。您可以从 [Pull secret](#) 下载 pull secret。如需有关 [pull secret](#) 的更多信息，请参阅使用镜像 pull secret。
4. 选择 **Add** 来创建您的凭证。

要创建使用此凭证的集群，您可以完成 [在内部环境中创建集群中的步骤](#)。

当您不再管理使用凭证的集群时，请删除凭证来保护凭证中的信息。选择要批量删除的 **Actions**，或者选择您要删除的凭证旁边的选项菜单。

1.5. 集群生命周期简介

multicluster engine operator 是集群生命周期 Operator，它为 OpenShift Container Platform 和 Red Hat

Advanced Cluster Management hub 集群提供集群管理功能。multicluster engine operator 是一个软件 Operator，它增强了集群管理，并支持跨云和数据中心的 OpenShift Container Platform 集群生命周期管理。您可以在没有 Red Hat Advanced Cluster Management 的情况下使用 multicluster engine operator。Red Hat Advanced Cluster Management 还自动安装多集群引擎 operator，并提供进一步的多集群功能。

请参阅以下文档：

- [集群生命周期架构](#)
- [管理凭证概述](#)
- [发行镜像](#)
 - [指定发行镜像](#)
 - [连接时维护自定义的发行镜像列表](#)
 - [断开连接时维护自定义的发行镜像列表](#)
- [主机清单简介](#)
- [创建集群](#)
 - [使用 CLI 创建集群](#)
 - [在集群创建过程中配置额外清单](#)
 - [在 Amazon Web Services 上创建集群](#)
 - [在 Amazon Web Services GovCloud 上创建集群](#)
 - [在 Microsoft Azure 上创建集群](#)
 - [在 Google Cloud Platform 上创建集群](#)
 - [在 VMware vSphere 上创建集群](#)
 - [在 Red Hat OpenStack Platform 上创建集群](#)
 - [在 Red Hat Virtualization 上创建集群](#)
 - [在内部环境中创建集群](#)
 - [在代理环境中创建集群](#)
 - [休眠创建的集群（技术预览）](#)
- [将目标受管集群导入到 hub 集群](#)
 - [使用控制台导入受管集群](#)
 - [使用 CLI 导入受管集群](#)
 - [在受管集群中指定镜像 registry 进行导入](#)
- [访问集群](#)

- 扩展受管集群
 - 使用 MachinePool 扩展
- 启用集群代理附加组件
- 配置 Ansible Automation Platform 任务以在受管集群上运行
- ClusterClaims
 - 列出现有 ClusterClaims
 - 创建自定义 ClusterClaims
- *ManagedClusterSets*
 - 创建 *ManagedClusterSet*
 - 为 *ManagedClusterSet* 分配 RBAC 权限
 - 创建 *ManagedClusterSetBinding* 资源
 - 使用污点和容忍放置受管集群
 - 从 *ManagedClusterSet* 中删除受管集群
- 在放置中使用 *ManagedClusterSet*
- 管理集群池（技术预览）
 - 创建集群池
 - 从集群池中声明集群
 - 更新集群池发行镜像
 - 扩展集群池
 - 销毁一个集群池
- 启用 *ManagedServiceAccount*
- 集群生命周期高级配置
- 从管理中移除集群

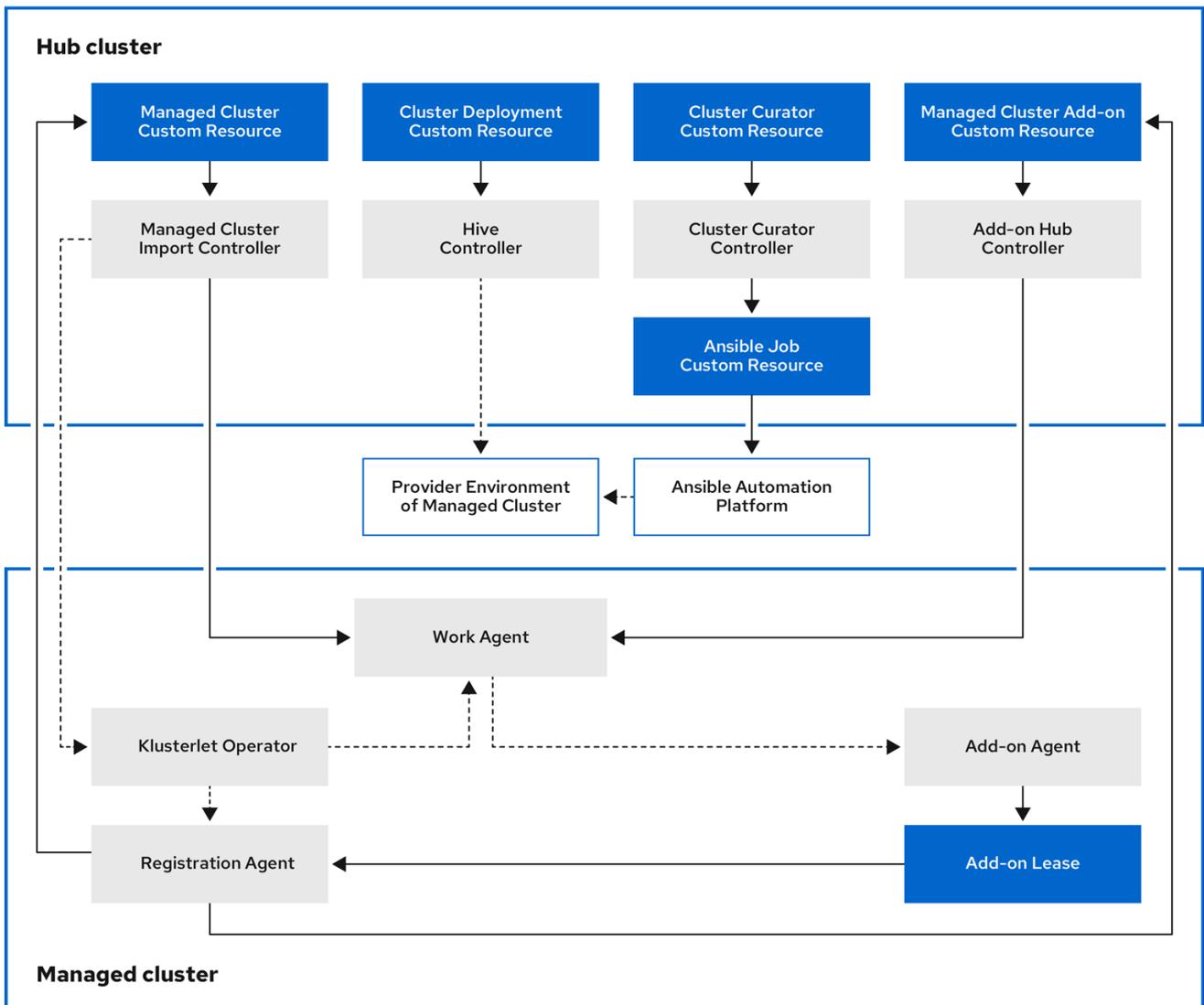
1.5.1. 集群生命周期架构

集群生命周期可以识别两种类型的集群：*hub clusters* 和 *managed clusters*。

hub 集群是带有 multicluster engine operator 的 OpenShift Container Platform（或 Red Hat Advanced Cluster Management）主集群。您可以使用 hub 集群创建、管理和监控其他 Kubernetes 集群。您可以使用 hub 集群创建集群，同时也可以导入现有集群由 hub 集群管理。

在创建受管集群时，集群会使用 Hive 资源的 Red Hat OpenShift Container Platform 集群安装程序创建集群。您可以通过阅读 OpenShift Container Platform 文档中的 OpenShift Container Platform 安装概述来了解更多有关使用 [OpenShift Container Platform 安装程序安装](#) 集群的过程的信息。

下图显示了为集群管理的 multicluster engine 安装的组件：



303_RHACM_0223

集群生命周期管理架构的组件包括以下项目：

1.5.1.1. hub 集群

- 受管集群导入控制器将 klusterlet Operator 部署到受管集群。
- Hive 控制器使用 Kubernetes operator 的多集群引擎置备您创建的集群。Hive Controller 还销毁由 Kubernetes operator 的多集群引擎创建的受管集群。
- 集群 curator 控制器将 Ansible 作业创建为 pre-hook 或 post-hook，以便在创建或升级受管集群时配置集群基础架构环境。
- 当在 hub 集群中启用了受管集群附加组件时，会在 hub 集群中部署其附加组件 hub 控制器。附加组件 hub 控制器将附加组件代理部署到受管集群。

1.5.1.2. 受管集群 (managed cluster)

- klusterlet Operator 在受管集群中部署注册和工作控制器。

- *Registration Agent* 将受管集群和受管集群附加组件注册到 hub 集群。*Registration Agent* 还维护受管集群和受管集群附加组件的状态。*Clusterrole* 中会自动创建以下权限，以允许受管集群访问 hub 集群：
 - 允许代理获取或更新 hub 集群管理的所有权集群
 - 允许代理更新 hub 集群管理的拥有集群状态
 - 允许代理轮转其证书
 - 允许代理 获取 或更新 **coordination.k8s.io** 租期
 - 允许代理 获取其 受管集群附加组件
 - 允许代理更新其受管集群附加组件的状态
- 工作代理将 Add-on Agent 应用到受管集群。允许受管集群访问 hub 集群的权限在 *Clusterrole* 中自动创建，并允许代理将事件发送到 hub 集群。

要继续添加和管理集群，请参阅 [集群生命周期简介](#)。

1.5.2. 发行镜像

当使用 *multicluster engine operator* 在供应商处创建集群时，您必须指定用于新集群的发行镜像。发行镜像指定使用哪个版本的 Red Hat OpenShift Container Platform 来构建集群。默认情况下，OpenShift Container Platform 使用 **clusterImageSets** 资源来获取支持的发行镜像列表。

[acm-hive-openshift-releases GitHub 仓库](#) 包含 OpenShift Container Platform 支持的 **clusterImageSets** 的 YAML 文件。此 Git 存储库的内容使用目录结构来根据 OpenShift Container Platform 版本和发行频道值(**fast,stable,candidate**)分隔镜像。Git 存储库中的分支映射到 OpenShift Container Platform 发行版本，每个分支都包含对应 OpenShift Container Platform 发行版本支持的 **clusterImageSets** YAML 文件。

在 *multicluster engine operator* 中，在 hub 集群上运行的集群镜像设置控制器。此控制器按设置间隔查询 [acm-hive-openshift-releases GitHub 仓库](#)，用于新的 **clusterImageSets** YAML 文件。默认情况下，控制器与 **backplane-2.2** 分支中的 **fast** 频道同步。

您可以使用以下选项配置 **ClusterImageSets**。

- 选项 1：指定要在控制台创建集群时使用的特定 **ClusterImageSet** 的镜像引用。您指定的每个新条目都会保留，并可用于将来的所有集群置备。请参见以下示例条目：

```
quay.io/openshift-release-dev/ocp-release:4.12.8-x86_64
```

- 选项 2：手动创建并应用来自 **acm-hive-openshift-releases** GitHub 仓库的 **ClusterImageSets** YAML 文件。
- 选项 3：遵循 [cluster-image-set-controller GitHub 仓库](#) 中的 **README.md**，启用对来自 fork 的 GitHub 仓库的 **ClusterImageSets** 的自动更新。

集群镜像集控制器可以配置为使用其他 Git 存储库同步 **ClusterImageSets**。控制器从 **multicluster-engine** 命名空间中的 **cluster-image-set-git-repo** ConfigMap 读取 Git 存储库配置。您可以使用此 ConfigMap 来暂停控制器同步 **ClusterImageSets**。这可以通过在 **gitRepoUrl** 字段中指定不存在的/invalid URL 来实现，如下所示。

```
apiVersion: v1
```

```

kind: ConfigMap
metadata:
  name: cluster-image-set-git-repo
  namespace: multicluster-engine
data:
  gitRepoUrl: https://github.com/stolostron/bad-acm-hive-openshift-releases.git
  gitRepoBranch: backplane-2.2
  gitRepoPath: clusterImageSets
  channel: fast

```

注：只有带有标签为：**visible: 'true'** 的发行镜像才可以在控制台中创建集群时选择。**ClusterImageSet** 资源中的此标签示例在以下内容中提供：

```

apiVersion: config.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.12.8-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.12.8-x86_64

```

存储了额外的发行镜像，但无法在控制台中看到。要查看所有可用的发行镜像，请运行 **kubectl get clusterimageset**。

继续阅读以了解更多有关发行镜像的信息：

- [指定发行镜像](#)
- [连接时维护自定义的发行镜像列表](#)
- [断开连接时维护自定义的发行镜像列表](#)

1.5.2.1. 指定发行镜像

当使用 multicluster engine operator 在供应商处创建集群时，您必须指定用于新集群的发行镜像。发行镜像指定使用哪个版本的 Red Hat OpenShift Container Platform 来构建集群。默认情况下，OpenShift Container Platform 使用 **clusterImageSets** 资源来获取支持的发行镜像列表。

[找到 ClusterImageSets配置 ClusterImageSets创建发行镜像来在不同构架中部署集群](#)

1.5.2.1.1. 查找 ClusterImageSets

引用发行镜像的文件是在 [acm-hive-openshift-releases GitHub 仓库](#) [GitHub 仓库](#) 中维护的 YAML 文件。Red Hat Advanced Cluster Management 使用这些文件在控制台中创建可用发行镜像的列表。这包括 OpenShift Container Platform 的最新快速频道镜像。

控制台仅显示三个 OpenShift Container Platform 最新版本的最新发行镜像。例如，您可能在控制台选项中看到以下发行镜像：

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

控制台中只有最新版本可促进创建带有最新发行镜像的集群。在某些情况下，您可能需要创建特定版本的集群，因此还会继续提供老版本。

注：在控制台中创建集群时，只有具有 **visible: 'true'** 标签的发行镜像才可以选择。**ClusterImageSet** 资源中的此标签示例在以下内容中提供：

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

存储了额外的发行镜像，但无法在控制台中看到。要查看所有可用发行镜像，请在 CLI 中运行 **kubectl get clusterimageset**。

存储库包含 **clusterImageSets** 目录，这是使用发行镜像时使用的目录。**clusterImageSets** 目录包含以下目录：

- **Fast**：包含引用每个受支持 OpenShift Container Platform 版本的最新发行镜像版本的文件。此目录中的发行镜像经过测试、验证和支持。
- **Releases**：包含引用每个 OpenShift Container Platform 版本（table、fast 和 candidate 频道）的所有发行镜像的文件。请注意：这些版本并没有经过测试并确定是稳定的。
 - **Stable**：包含引用每个受支持 OpenShift Container Platform 版本的最新两个稳定发行镜像版本的文件。

注：默认情况下，当前发行镜像列表被更新一次。升级产品后，列表最多可能需要一小时，以反映该产品的新版本的建议发行镜像版本。

1.5.2.1.2. 配置 *ClusterImageSets*

您可以使用以下选项配置 **ClusterImageSets**。

- **选项 1**：指定要在控制台创建集群时使用的特定 **ClusterImageSet** 的镜像引用。您指定的每个新条目都会保留，并可用于将来的所有集群置备。请参见以下示例条目：

```
quay.io/openshift-release-dev/ocp-release:4.6.8-x86_64
```

- **选项 2**：手动创建并应用来自 **acm-hive-openshift-releases** GitHub 仓库的 **ClusterImageSets** YAML 文件。
- **选项 3**：遵循 [cluster-image-set-controller](#) GitHub 仓库中的 **README.md**，启用对来自 fork 的 GitHub 仓库的 **ClusterImageSets** 的自动更新。

1.5.2.1.3. 创建发行镜像以在不同构架中部署集群

您可以通过手动创建包含这两个架构文件的发行镜像，在与 hub 集群架构不同的架构中创建集群。

例如，您可以从 **ppc64le**、**aarch64** 或 **s390x** 架构上运行的 hub 集群创建一个 **x86_64** 集群。如果使用两组文件创建发行镜像，集群创建成功，因为新发行镜像启用 OpenShift Container Platform 发行 registry 来提供多架构镜像清单。

OpenShift Container Platform 4.11 及更新的版本默认支持多个架构。您可以使用以下 **clusterImageSet** 来置备集群：

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.12.0-multi-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.12.0-multi
```

要为不支持多个架构的 OpenShift Container Platform 镜像创建发行镜像，请完成类似以下架构类型的步骤：

1. 在 [OpenShift Container Platform release registry](#) 中，创建一个 [清单列表](#)，其中包含 **x86_64**、**s390x**、**aarch64** 和 **ppc64le** 发行镜像。

- a. 使用以下命令，从 [Quay 仓库](#) 拉取环境中这两个架构的清单列表：

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

- b. 登录到维护镜像的私有存储库：

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

使用存储库的路径替换 **private-repo**。

- c. 运行以下命令，将发行镜像清单添加到私有存储库中：

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

使用存储库的路径替换 **private-repo**。

- d. 为新信息创建清单：

```
podman manifest create mymanifest
```

- e. 将两个发行镜像的引用添加到清单列表中：

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

- 使用存储库的路径替换 **private-repo**。

- f. 将清单列表中的列表与现有清单合并：

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

- 使用存储库的路径替换 **private-repo**。

2. 在 hub 集群中，创建一个发行版本镜像来引用存储库中的清单。

- a. 创建一个 YAML 文件，其中包含类似以下示例的信息：

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

- 使用存储库的路径替换 **private-repo**。

- b. 在 hub 集群中运行以下命令以应用更改：

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

- 将 **file-name** 替换为您刚才创建的 YAML 文件的名称。

3. 在创建 OpenShift Container Platform 集群时选择新的发行镜像。
4. 如果使用 Red Hat Advanced Cluster Management 控制台部署受管集群，在集群创建过程中在 *Architecture* 字段中指定受管集群的架构。

创建流程使用合并的发行镜像来创建集群。

1.5.2.2. 连接时维护自定义的发行镜像列表

您可能想要在所有集群中使用相同的发行镜像。为简化操作，您可以创建自己的自定义列表，在其中包含创建集群时可用的发行镜像。完成以下步骤以管理可用发行镜像：

1. 对 [acm-hive-openshift-releases GitHub 仓库 2.7 分支进行分叉\(fork\)](#)。
2. 为创建集群时可用的镜像添加 YAML 文件。使用 Git 控制台或终端将镜像添加到 `./clusterImageSets/stable/` 或 `./clusterImageSets/fast/` 目录中。
3. 在 **multicluster-engine** 命名空间中创建一个名为 **cluster-image-set-git-repo** 的 **ConfigMap**。请参见以下示例：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-image-set-git-repo
  namespace: multicluster-engine
```

```
data:
  gitRepoUrl: <forked acm-hive-openshift-releases repository URL>
  gitRepoBranch: backplane-2.2
  gitRepoPath: clusterImageSets
  channel: <fast or stable>
```

您可以按照以下流程将更改合并到已分叉的存储库，从主存储库检索可用的 YAML 文件：

1. 将更改提交并合并到您的已分叉仓库。
2. 要在克隆 **acm-hive-openshift-releases** 仓库后同步 fast 发行镜像列表，请将 **cluster-image-set-git-repo ConfigMap** 中的 channel 字段的值更新为 **fast**。
3. 要同步并显示 stable 发行镜像，请将 **cluster-image-set-git-repo ConfigMap** 中的 channel 字段的值更新为 **stable**。

更新 **ConfigMap** 后，可用稳定镜像列表会在约一分钟内更新为当前可用镜像。默认情况下，Red Hat Advanced Cluster Management 会预加载几个 **ClusterImageSets**。

1. 您可以使用以下命令列出可用内容并删除默认值。将 **<clusterImageSet_NAME>** 替换为正确的名称：

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

在创建集群时，在控制台中查看当前可用发行镜像的列表。

有关通过 **ConfigMap** 提供的其他字段的信息，请查看 [cluster-image-set-controller GitHub 仓库 README](#)。

1.5.2.3. 断开连接时维护自定义的发行镜像列表

在某些情况下，当节点集群没有互联网连接时，您需要维护一个自定义的发行镜像列表。您可以创建自己的自定义列表，在其中包含创建集群时可用的发行镜像。完成以下步骤以在断开连接的情况下管理可用发行镜像：

1. 在连接的系统中时，进入 [acm-hive-openshift-releases GitHub 仓库](#)，以访问可用于版本 2.7 的集群镜像集。
2. 将 **clusterImageSets** 目录复制到可以访问断开连接的多集群引擎 operator 集群的系统中。
3. 通过完成以下步骤，添加受管集群和带有集群镜像的断开连接的存储库之间的映射：
 - 对于 OpenShift Container Platform 受管集群，请参阅[配置镜像 registry 存储库镜像](#)，以了解有关使用 **ImageContentSourcePolicy** 对象完成映射的信息。
 - 对于不是 OpenShift Container Platform 集群的受管集群，请使用 **ManageClusterImageRegistry** 自定义资源定义来覆盖镜像集的位置。如需有关如何 [为映射覆盖集群的信息](#)，请参阅[指定受管集群中的 registry 镜像](#)。
4. 使用控制台或 CLI 为您希望在创建集群时可用的镜像添加 YAML 文件，以手动添加 **clusterImageSet** YAML 内容。
5. 修改 OpenShift Container Platform 发行镜像的 **clusterImageSet** YAML 文件，以引用存储镜像的正确离线存储库。您的更新类似以下示例，其中 **spec.releaseImage** 指的是您要使用的镜像 registry：

-

```

apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    name: img4.12.8-x86-64-appsub
spec:
  releaseImage: IMAGE_REGISTRY_IPADDRESS_or_DNSNAME/REPO_PATH/ocp-
  release:4.12.8-x86_64

```

确保在 YAML 文件中引用的离线镜像 registry 中载入镜像。

6. 通过为每个 **YAML** 文件输入以下命令来创建每个 clusterImageSets :

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

将 **clusterImageSet_FILE** 替换为集群镜像集文件的名称。例如 :

```
oc create -f img4.11.9-x86_64.yaml
```

为要添加的每个资源运行此命令后，可用发行镜像列表可用。

7. 另外，您还可以将镜像 URL 直接粘贴到创建集群控制台中。如果镜像 URL 不存在，则添加镜像 URL 会创建新的 **clusterImageSets**。
8. 在创建集群时，在控制台中查看当前可用发行镜像的列表。

1.5.3. 主机清单简介

主机清单管理和内部集群安装可以使用 multicluster engine operator 中央基础架构管理功能。中央基础架构管理作为 hub 集群上的 operator 运行 Assisted Installer（也称为基础架构 operator）。

您可以使用控制台创建主机清单，这是可用于创建内部 OpenShift Container Platform 集群的裸机或虚拟机池。这些集群可以独立使用 control plane 或 **托管的** control plane 的专用机器，其中 control plane 作为 hub 集群上的 pod 运行。

您可以使用 Zero Touch Provisioning (ZTP) 使用 console、API 或 GitOps 安装独立集群。如需有关 ZTP 的更多信息，请参阅 [Red Hat OpenShift Container Platform 文档中的在断开连接的环境中安装 GitOps ZTP](#)。

机器在发现镜像引导后加入主机清单。Discovery Image 是一个包含以下内容的 Red Hat CoreOS live 镜像：

- 执行发现、验证和安装任务的代理。
- 用于访问 hub 集群上的服务所需的配置，包括端点、令牌和静态网络配置（如果适用）。

您通常为每个基础架构环境有一个发现镜像，这是一组共享一组通用属性的主机。**InfraEnv** 自定义资源定义代表此基础架构环境和关联的发现镜像。使用的镜像基于 OpenShift Container Platform 版本，该版本决定了所选的操作系统版本。

主机引导且代理联系该服务后，服务会在代表该主机的 hub 集群上创建一个新的 **Agent** 自定义资源。**Agent** 资源组成主机清单。

稍后，您可以在清单中安装主机作为 OpenShift 节点。代理将操作系统写入磁盘，以及必要的配置，并重新启动主机。

继续阅读以了解更多有关主机清单和中央基础架构管理的信息：

- [启用中央基础架构管理服务](#)
- [在 Amazon Web Services 上启用中央基础架构管理](#)
- [使用控制台创建主机清单](#)
- [使用命令行界面创建主机清单](#)
- [为基础架构环境配置高级网络](#)
- [使用 Discovery 镜像将主机添加到主机清单中](#)
- [自动将裸机主机添加到主机清单中](#)
- [管理主机清单](#)
- [在内部环境中创建集群](#)

1.5.3.1. 启用中央基础架构管理服务

中央基础架构管理服务由 multicluster engine operator 提供，并部署 OpenShift Container Platform 集群。当您在 hub 集群上启用 MultiClusterHub Operator 时，中央基础架构管理会自动部署，但您必须手动启用该服务。

1.5.3.1.1. 先决条件

在启用中央基础架构管理服务前，请查看以下先决条件：

- 您必须在 OpenShift Container Platform 4.10 或更高版本上部署了 hub 集群，并带有支持的 Red Hat Advanced Cluster Management for Kubernetes 版本。
- 您需要对 hub 集群（连接）或连接到连接到互联网（断开连接）的内部或镜像 registry 的连接，以检索创建环境所需的镜像。
- 您必须为裸机置备打开所需的端口。请参阅 OpenShift Container Platform 文档中的 [确保开放所需端口](#)。
- 您需要裸机主机自定义资源定义。
- 您需要 OpenShift Container Platform [pull secret](#)。如需更多信息，请参阅[使用镜像 pull secret](#)。
- 您需要一个配置的默认存储类。
- 对于断开连接的环境，在 OpenShift Container Platform [文档中的网络边缘](#) 完成集群的步骤。

1.5.3.1.2. 创建裸机主机自定义资源定义

在启用中央基础架构管理服务前，您需要裸机主机自定义资源定义。

1. 运行以下命令，检查是否已有一个裸机主机自定义资源定义：

```
oc get crd baremetalhosts.metal3.io
```

- 如果您有裸机主机自定义资源定义，输出会显示创建资源的日期。

- 如果您没有资源，您会收到类似如下的错误：

```
Error from server (NotFound): customresourcedefinitions.apiextensions.k8s.io
"baremetalhosts.metal3.io" not found
```

2. 如果您没有裸机主机自定义资源定义，请下载 [metal3.io_baremetalhosts.yaml](#) 文件，并通过运行以下命令来应用内容来创建资源：

```
oc apply -f
```

1.5.3.1.3. 创建或修改 置备 资源

在启用中央基础架构管理服务前，您需要 置备 资源。

1. 运行以下命令，检查您是否有 **Provisioning** 资源：

```
oc get provisioning
```

- 如果您已经有一个 **Provisioning** 资源，请继续 *修改 Provisioning 资源*。
- 如果您没有 **Provisioning** 资源，您会收到 **No resources found** 错误。继续 *创建 Provisioning 资源*。

1.5.3.1.3.1. 修改 置备 资源

如果您已经有一个 **Provisioning** 资源，则必须在以下平台之一上安装了 hub 集群，修改资源：

- 裸机
- Red Hat OpenStack Platform
- VMware vSphere
- 用户置备的基础架构(UPI)方法，平台为 **None**

如果您的 hub 集群安装在不同的平台上，请继续 *在断开连接的环境中启用中央基础架构管理*，或者在 *连接的环境中启用中央基础架构管理*。

1. 运行以下命令，修改 **Provisioning** 资源以允许 Bare Metal Operator 监视所有命名空间：

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

1.5.3.1.3.2. 创建 置备 资源

如果您没有 **Provisioning** 资源，请完成以下步骤：

1. 通过添加以下 YAML 内容来创建 **Provisioning** 资源：

```
apiVersion: metal3.io/v1alpha1
kind: Provisioning
metadata:
  name: provisioning-configuration
```

```
spec:
  provisioningNetwork: "Disabled"
  watchAllNamespaces: true
```

2. 运行以下命令来应用内容：

```
oc apply -f
```

1.5.3.1.4. 在断开连接的环境中启用中央基础架构管理

要在断开连接的环境中启用中央基础架构管理，请完成以下步骤：

1. 在与基础架构 Operator 相同的命名空间中创建 **ConfigMap**，为您的镜像 registry 指定 **ca-bundle.crt** 和 **registry.conf** 的值。您的文件 **ConfigMap** 可能类似以下示例：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: <mirror-config>
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    <certificate-content>
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
    [[registry]]
      prefix = ""
      location = "registry.redhat.io/multicluster-engine"
      mirror-by-digest-only = true
    [[registry.mirror]]
      location = "mirror.registry.com:5000/multicluster-engine"
```

unqualified-search-registries 列表中的 registry 会自动添加到 **PUBLIC_CONTAINER_REGISTRIES** 环境变量中的身份验证忽略列表中。当验证受管集群的 pull secret 时，指定的 registry 不需要身份验证。

2. 通过在 **agent_service_config.yaml** 文件中保存以下 YAML 内容来创建 **AgentServiceConfig** 自定义资源：

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

```

resources:
  requests:
    storage: <fs_volume_size>
mirrorRegistryRef:
  name: <mirror_config> ❶
unauthenticatedRegistries:
  - <unauthenticated_registry> ❷
imageStorage:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: <img_volume_size> ❸
osImages:
  - openshiftVersion: "<ocp_version>" ❹
    version: "<ocp_release_version>" ❺
    url: "<iso_url>" ❻
    cpuArchitecture: "x86_64"

```

- ❶ 将 **mirror_config** 替换为包含您的镜像 registry 配置详情的 **ConfigMap** 名称。
- ❷ 如果您使用不需要身份验证的镜像 registry，请包含可选的 **unauthenticated_registry** 参数。此列表上的条目不会被验证，或者需要在 pull secret 中有一个条目。
- ❸ 将 **img_volume_size** 替换为 **imageStorage** 字段的卷大小，如每个操作系统镜像的 **10Gi**。最小值为 **10Gi**，但建议的值至少为 **50Gi**。这个值指定为集群镜像分配多少存储。您需要为每个运行的 Red Hat Enterprise Linux CoreOS 实例提供 1 GB 的镜像存储。如果 Red Hat Enterprise Linux CoreOS 有多个集群和实例，您可能需要使用更高的值。
- ❹ 将 **ocp_version** 替换为要安装的 OpenShift Container Platform 版本，如 **4.13**。
- ❺ 将 **ocp_release_version** 替换为特定的安装版本，例如：**49.83.202103251640-0**。
- ❻ 使用 ISO url 替换 **iso_url**，例如 https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/rhcos-4.10.3-x86_64-live.x86_64.iso。您可以在 [4.10.3 依赖项](#) 中找到其他值。

重要：如果您使用更新的绑定功能以及 **AgentServiceConfig** 自定义资源中的 **spec.osImages** 发行版本是 4.13 或更高版本，则您在创建集群时使用的 OpenShift Container Platform 发行镜像必须是 4.13 或更高版本。版本 4.13 及更新版本的 Red Hat Enterprise Linux CoreOS 镜像与版本 4.13 之前的版本不兼容。

您可以通过检查 **assisted-service** 和 **assisted-image-service** 部署，并验证您的中央基础架构管理服务是否健康，并确保其 pod 已准备就绪并在运行。

1.5.3.1.5. 在连接的环境中启用中央基础架构管理

要在连接的环境中启用中央基础架构管理，通过在 **agent_service_config.yaml** 文件中保存以下 YAML 内容来创建 **AgentServiceConfig** 自定义资源：

```

apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  name: agent
spec:

```

```

databaseStorage:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: <db_volume_size> 1
filesystemStorage:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: <fs_volume_size> 2
imageStorage:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: <img_volume_size> 3

```

- 1 使用 **databaseStorage** 字段的卷大小替换 **db_volume_size**，如 **10Gi**。这个值指定为存储集群分配的存储量，如数据库表和数据库视图。所需的最小值为 **1Gi**。如果有多个集群，您可能需要使用较高的值。
- 2 将 **fs_volume_size** 替换为 **filesystemStorage** 字段的卷大小，例如，每个集群 **200M** 和每个支持的 OpenShift Container Platform 版本 **2-3Gi**。所需的最小值为 **1Gi**，但推荐的值为至少 **100Gi**。这个值指定为存储集群的日志、清单和 **kubeconfig** 文件分配了多少存储。如果有多个集群，您可能需要使用较高的值。
- 3 将 **img_volume_size** 替换为 **imageStorage** 字段的卷大小，如每个操作系统镜像的 **10Gi**。最小值为 **10Gi**，但建议的值至少为 **50Gi**。这个值指定为集群镜像分配多少存储。您需要为每个运行的 Red Hat Enterprise Linux CoreOS 实例提供 1 GB 的镜像存储。如果 Red Hat Enterprise Linux CoreOS 有多个集群和实例，您可能需要使用更高的值。

您的中央基础架构管理服务已配置。您可以通过检查 **assisted-service** 和 **assisted-image-service** 部署，确定 pod 已就绪并在运行，来验证其状态是否正常。

1.5.3.1.6. 其他资源

- 如需有关零接触置备的更多信息，请参阅 OpenShift Container Platform 文档中的 [网络边缘集群](#)。
- [请参阅使用镜像 pull secret](#)

1.5.3.2. 在 Amazon Web Services 上启用中央基础架构管理

如果您在 Amazon Web Services 上运行 hub 集群并希望启用中央基础架构管理服务，请在 [启用中央基础架构管理服务](#) 后完成以下步骤：

1. 确保您已在 hub 集群中登录，并通过运行以下命令来查找在 **assisted-image-service** 上配置的唯一域：

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

您的域可能类似以下示例：**assisted-image-service-multicluster-engine.apps.<yourdomain>.com**

2. 确保您已在 hub 集群中登录，并使用 **NLB type** 参数创建带有唯一域的新 **IngressController**。请参见以下示例：

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: ingress-controller-with-nlb
  namespace: openshift-ingress-operator
spec:
  domain: nlb-apps.<domain>.com
  routeSelector:
    matchLabels:
      router-type: nlb
  endpointPublishingStrategy:
    type: LoadBalancerService
  loadBalancer:
    scope: External
  providerParameters:
    type: AWS
    aws:
      type: NLB
```

3. 将 **<yourdomain>** 添加到 **IngressController** 中的 **domain** 参数，方法是使用 **<yourdomain>** 替换 **nlb-apps.<domain>.com** 的 **<domain>**。
4. 运行以下命令来应用新的 **IngressController**：

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

5. 通过完成以下步骤，确保新 **IngressController** 的 **spec.domain** 参数的值不与现有 **IngressController** 冲突：
 - a. 运行以下命令列出所有 **IngressController**：

```
oc get ingresscontroller -n openshift-ingress-operator
```

- b. 在每个 **IngressController** 上运行以下命令，除了您刚才创建的 **ingress-controller-with-nlb** 外：

```
oc edit ingresscontroller <name> -n openshift-ingress-operator
```

如果缺少 **spec.domain** 报告，请添加一个与集群中公开的所有路由匹配的默认域，但 **nlb-apps.<domain>.com** 除外。

如果提供了 **spec.domain** 报告，请确保从指定范围中排除 **nlb-apps.<domain>.com** 路由。

6. 运行以下命令来编辑 **assisted-image-service** 路由以使用 **nlb-apps** 位置：

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

默认命名空间是您安装 multicluster engine operator 的位置。

- 在 **assisted-image-service** 路由中添加以下行：

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

- 在 **assisted-image-service** 路由中，找到 **spec.host** 的 URL 值。URL 可能类似以下示例：

```
assisted-image-service-multicluster-engine.apps.<yourdomain>.com
```

- 将 URL 中的 **apps** 替换为 **nlb-apps**，以匹配新 **IngressController** 中配置的域。
- 要验证 Amazon Web Services 上是否启用了中央基础架构管理服务，请运行以下命令来验证 pod 是否健康：

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

- 创建新主机清单，并确保下载 URL 使用新的 **nlb-apps** URL。

1.5.3.3. 使用控制台创建主机清单

您可以创建一个主机清单（基础架构环境）来发现您可以在其上安装 OpenShift Container Platform 集群的物理或虚拟集群。

1.5.3.3.1. 先决条件

- 您必须启用中央基础架构管理服务。如需更多信息，请参阅[启用中央基础架构管理服务](#)。

1.5.3.3.2. 创建主机清单

完成以下步骤，使用控制台创建主机清单：

- 在控制台中，导航到 **Infrastructure > Host inventory**，再点 **Create infrastructure environment**。
- 在主机清单设置中添加以下信息：
 - 名称**：您的基础架构环境的唯一名称。使用控制台创建基础架构环境也会使用您选择的名称为 **InfraEnv** 资源创建新命名空间。如果您使用命令行界面创建 **InfraEnv** 资源并希望您在控制台中监控资源，请为您的命名空间和 **InfraEnv** 使用相同的名称。
 - Network type**：指定您添加到基础架构环境中的主机是否使用 DHCP 或静态网络。静态网络配置需要额外的步骤。
 - location**：指定主机的地理位置。地理位置可用于定义主机所在的数据中心。
 - Labels**：可选字段，您可以在其中向使用此基础架构环境发现的主机添加标签。指定的位置会自动添加到标签列表中。
 - 基础架构供应商凭证**：选择基础架构供应商凭证会使用凭证中的信息自动填充 pull secret 和 SSH 公钥字段。如需更多信息，请参阅[为内部环境创建凭证](#)。
 - pull secret**：用于访问 OpenShift Container Platform 资源的 OpenShift Container Platform [pull secret](#)。如果您选择了一个基础架构供应商凭证，则会自动填充此字段。

- **SSH 公钥**：实现与主机安全通信的 SSH 密钥。您可以使用它来连接到主机以进行故障排除。安装集群后，无法再使用 SSH 密钥连接到主机。密钥通常位于您的 `id_rsa.pub` 文件中。默认文件路径为 `~/.ssh/id_rsa.pub`。如果您选择了包含 SSH 公钥值的基础架构供应商凭证，则会自动填充此字段。
- 如果要为主机启用代理设置，请选择要启用它的设置，并输入以下信息：
 - **HTTP 代理 URL**：HTTP 请求的代理 URL。
 - **HTTPS 代理 URL**：HTTP 请求的代理 URL。URL 必须以 HTTP 开头。不支持 HTTPS。如果没有提供值，则默认将 HTTP 代理 URL 用于 HTTP 和 HTTPS 连接。
 - **无代理域**：您不想将代理与逗号分开的域列表。使用一个句点(.)启动域名，以包含该域中的所有子域。添加一个星号(*)来绕过所有目的地的代理。
- (可选) 通过提供以逗号分隔的 NTP 池或服务器的 IP 或域名列表来添加您自己的网络时间协议(NTP)源。

如果您需要控制台中不可用的高级配置选项，请使用命令行界面继续 [创建主机清单](#)。

如果您不需要高级配置选项，则可以继续配置静态网络（如果需要），并开始将主机添加到基础架构环境中。

1.5.3.3.3. 访问主机清单

要访问主机清单，请在控制台中选择 **Infrastructure > Host inventory**。从列表中选择您的基础架构环境，以查看详情和主机。

1.5.3.3.4. 其他资源

- [请参阅启用中央基础架构管理服务](#)
- [请参阅为内部环境创建凭证](#)

1.5.3.4. 使用命令行界面创建主机清单

您可以创建一个主机清单（基础架构环境）来发现您可以在其上安装 OpenShift Container Platform 集群的物理或虚拟集群。使用命令行界面而不是控制台进行自动化部署或以下高级配置选项：

- 自动将发现的主机绑定到现有集群定义
- 覆盖发现镜像的 ignition 配置
- 控制 iPXE 行为
- 修改发现镜像的内核参数
- 在发现阶段传递您希望主机信任的额外证书
- 选择要引导的 Red Hat CoreOS 版本，用于测试不是最新版本的默认选项

1.5.3.4.1. 前提条件

- 您必须启用中央基础架构管理服务。如需更多信息，[请参阅启用中央基础架构管理服务](#)。

1.5.3.4.2. 创建主机清单

完成以下步骤，使用命令行界面创建主机清单（基础架构环境）：

1. 运行以下命令登录到您的 hub 集群：

```
oc login
```

2. 为资源创建命名空间。
 - a. 创建 **namespace.yaml** 文件并添加以下内容：

```
apiVersion: v1
kind: Namespace
metadata:
  name: <your_namespace> ①
```

- ① 在控制台中使用命名空间和基础架构环境相同的名称来监控您的清单。

- b. 运行以下命令来应用 YAML 内容：

```
oc apply -f namespace.yaml
```

3. 创建包含 OpenShift Container Platform [pull secret](#) 的 **Secret** 自定义资源。

- a. 创建 **pull-secret.yaml** 文件并添加以下内容：

```
apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: pull-secret ①
  namespace: <your_namespace>
stringData:
  .dockerconfigjson: <your_pull_secret> ②
```

- ① 添加您的名称space。

- ② 添加 pull secret。

- b. 运行以下命令来应用 YAML 内容：

```
oc apply -f pull-secret.yaml
```

4. 创建基础架构环境。

- a. 创建 **infra-env.yaml** 文件并添加以下内容。根据需要替换值：

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: myinfraenv
  namespace: <your_namespace>
spec:
  proxy:
```

```

httpProxy: <http://user:password@ipaddr:port>
httpsProxy: <http://user:password@ipaddr:port>
noProxy:
additionalNTPSources:
sshAuthorizedKey:
pullSecretRef:
  name: <name>
agentLabels:
  <key>: <value>
nmStateConfigLabelSelector:
  matchLabels:
    <key>: <value>
clusterRef:
  name: <cluster_name>
  namespace: <project_name>
ignitionConfigOverride: '{"ignition": {"version": "3.1.0"}, ...}'
cpuArchitecture: x86_64
ipxeScriptType: DiscoveryImageAlways
kernelArguments:
  - operation: append
    value: audit=0
additionalTrustBundle: <bundle>
osImageVersion: <version>

```

表 1.3. InfraEnv 字段表

字段	可选或必需的	描述
proxy	选填	定义使用 InfraEnv 资源的代理和集群的代理设置。如果没有设置 proxy 值，则代理不会配置为使用代理。
httpProxy	选填	HTTP 请求的代理的 URL。URL 必须以 http 开头。不支持 HTTPS。
httpsProxy	选填	HTTP 请求的代理的 URL。URL 必须以 http 开头。不支持 HTTPS。
noProxy	选填	您不想将代理与逗号分开的域和 CIDR 列表。
additionalNTPSources	选填	要添加到所有主机的网络时间协议 (NTP) 源 (主机名或 IP) 的列表。它们添加到使用其他选项 (如 DHCP) 配置的 NTP 源中。
sshAuthorizedKey	选填	添加到所有主机的 SSH 公钥，以便在发现阶段进行调试。发现阶段是主机引导发现镜像时。
name	必需	包含 pull secret 的 Kubernetes secret 名称。

字段	可选或必需的	描述
agentLabels	选填	自动添加到代理资源的标签，代表您的 InfraEnv 发现的主机。确保添加您的键和值。
nmStateConfigLabelSelector	选填	整合主机的高级网络配置，如静态 IP、网桥和绑定。主机网络配置在一个或多个 NMStateConfig 资源中指定，使用您选择的标签。 nmStateConfigLabelSelector 属性是一个 Kubernetes 标签选择器，它与您选择的标签匹配。与此标签选择器匹配的所有 NMStateConfig 标签的网络配置都包含在 Discovery Image 中。当您引导时，每个主机会将每个配置与其网络接口进行比较，并应用适当的配置。要了解更多有关高级网络配置的信息，请参阅 为主机清单配置高级网络 的链接。
clusterRef	选填	引用描述独立内部集群的现有 ClusterDeployment 资源。默认不设置。如果没有设置 clusterRef ，则主机稍后可以绑定到一个或多个集群。您可以从一个集群中删除主机并将其添加到另一个集群中。如果设置了 clusterRef ，则通过 InfraEnv 发现的所有主机都会自动绑定到指定的集群。如果集群还没有安装，则所有发现的主机都属于其安装的一部分。如果已安装集群，则会添加所有发现的主机。
ignitionConfigOverride	选填	修改 Red Hat CoreOS live 镜像的 ignition 配置，如添加文件。如果需要，请确保只使用 ignitionConfigOverride 。必须使用 ignition 版本 3.1.0，无论集群版本是什么。
cpuArchitecture	选填	选择以下支持的 CPU 架构之一：x86_64、aarch64、ppc64le 或 s390x。默认值为 x86_64。

字段	可选或必需的	描述
ipxeScriptType	选填	当设置为 DiscoveryImageAlways 的默认值以及使用 iPXE 引导时，导致镜像服务始终提供 iPXE 脚本。因此，主机从网络发现镜像引导。将值设为 BootOrderControl 会导致镜像服务决定何时返回 iPXE 脚本，具体取决于主机状态，这会导致主机在主机调配并且是集群的一部分时从磁盘引导。
kernelArguments	选填	允许在发现镜像引导时修改内核参数。操作可能的值有附加、 replace 或删除。
additionalTrustBundle	选填	如果主机位于带有重新加密 man-in-the-middle (MITM)代理的网络中，或者主机需要信任证书进行其他目的，则需要 PEM 编码的 X.509 证书捆绑包。 InfraEnv 发现的主机信任此捆绑包中的证书。从 InfraEnv 发现的主机创建的集群也会信任此捆绑包中的证书。
osImageVersion	选填	用于 InfraEnv 的 Red Hat CoreOS 镜像版本。确保版本引用 AgentServiceConfig.spec.osImages 或默认 OS 镜像列表中指定的 OS 镜像。每个发行版本都有一组特定的 Red Hat CoreOS 镜像版本。 OSImageVersion 必须与 OS 镜像列表中的 OpenShift Container Platform 版本匹配。您不能同时指定 OSImageVersion 和 ClusterRef 。如果要使用默认不存在的 Red Hat CoreOS 镜像的另一个版本，则必须在 AgentServiceConfig.spec.osImages 中指定版本来手动添加版本。要了解更多信息有关添加版本的信息，请参阅 启用中央基础架构管理服务 。

- a. 运行以下命令来应用 YAML 内容：

```
oc apply -f infra-env.yaml
```

- b. 要验证您的主机清单是否已创建，请使用以下命令检查状态：

-

```
oc describe infraenv myinfraenv -n <your_namespace>
```

请参阅以下显著属性列表：

- **条件**：指示镜像是否被成功创建的标准 Kubernetes 条件。
- **isoDownloadURL**：下载发现镜像的 URL。
- **createdTime**：镜像最后一次创建的时间。如果您修改了 **InfraEnv**，请确保在下载新镜像前更新时间戳。

注：如果您修改 **InfraEnv** 资源，请通过查看 **createdTime** 属性来确保 **InfraEnv** 已创建新的 Discovery 镜像。如果您已经引导主机，请使用最新的发现镜像再次引导它们。

如果需要，可以继续配置静态网络，并开始将主机添加到基础架构环境中。

1.5.3.4.3. 其他资源

- [请参阅启用中央基础架构管理服务](#)
- 使用控制台返回到 [创建主机清单](#)

1.5.3.5. 为基础架构环境配置高级网络

对于在单个接口上需要 DHCP 之外的网络的主机，您必须配置高级网络。所需的配置包括创建一个或多个 **NMStateConfig** 资源实例，用于描述一个或多个主机的网络。

每个 **NMStateConfig** 资源都必须包含一个与 **InfraEnv** 资源中的 **nmStateConfigLabelSelector** 匹配的标签。请参阅[使用命令行界面创建主机清单](#)，以了解更多有关 **nmStateConfigLabelSelector** 的信息。

Discovery Image 包含在所有引用的 **NMStateConfig** 资源中定义的网络配置。引导后，每个主机会将每个配置与其网络接口进行比较，并应用适当的配置。

1.5.3.5.1. 先决条件

- 您必须启用中央基础架构管理服务。
- 您必须创建主机清单。

1.5.3.5.2. 使用命令行界面配置高级网络

要使用命令行界面为您的基础架构环境配置高级网络，请完成以下步骤：

1. 创建名为 **nmstateconfig.yaml** 的文件，并添加类似于以下模板的内容。根据需要替换值：

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: mynmstateconfig
  namespace: <your-infraenv-namespace>
  labels:
    some-key: <some-value>
spec:
  config:
    interfaces:
```

```

- name: eth0
  type: ethernet
  state: up
  mac-address: 02:00:00:80:12:14
  ipv4:
    enabled: true
    address:
      - ip: 192.168.111.30
        prefix-length: 24
    dhcp: false
- name: eth1
  type: ethernet
  state: up
  mac-address: 02:00:00:80:12:15
  ipv4:
    enabled: true
    address:
      - ip: 192.168.140.30
        prefix-length: 24
    dhcp: false
dns-resolver:
  config:
    server:
      - 192.168.126.1
routes:
  config:
    - destination: 0.0.0.0/0
      next-hop-address: 192.168.111.1
      next-hop-interface: eth1
      table-id: 254
    - destination: 0.0.0.0/0
      next-hop-address: 192.168.140.1
      next-hop-interface: eth1
      table-id: 254
interfaces:
  - name: "eth0"
    macAddress: "02:00:00:80:12:14"
  - name: "eth1"
    macAddress: "02:00:00:80:12:15"

```

表 1.4. NMStateConfig 字段表

字段	可选或必需的	描述
name	必需	使用与您要配置的主机或主机相关的名称。
namespace	必需	命名空间必须与 InfraEnv 资源的命名空间匹配。
some-key	必需	在 InfraEnv 资源中添加一个或多个与 nmStateConfigLabelSelector 匹配的标签。

字段	可选或必需的	描述
config	选填	描述 NMstate 格式的网络设置。有关格式规格和其他示例，请参阅 <i>Declarative Network API</i> 。配置也可以应用到单个主机，其中每个主机都有一个 NMStateConfig 资源，也可以描述单个 NMStateConfig 资源中的多个主机的接口。
interfaces	选填	描述主机上指定 NMstate 配置和 MAC 地址中找到的接口名称之间的映射。确保映射使用主机上存在的物理接口。例如，当 NMState 配置定义了绑定或 VLAN 时，映射仅包含父接口的条目。映射有以下目的：* 允许您在配置中使用与主机上的接口名称不匹配的接口名称。您可能会发现此功能很有用，因为操作系统选择接口名称，这可能不可预测。* 告知主机在引导后查找哪些 MAC 地址并应用正确的 NMstate 配置。

注：当您更新任何 **InfraEnv** 属性或更改与其标签选择器匹配的 **NMStateConfig** 资源时，Image Service 会自动创建一个新镜像。如果在创建 **InfraEnv** 资源后添加 **NMStateConfig** 资源，请通过检查 **InfraEnv** 中的 **createdTime** 属性来确保 **InfraEnv** 创建一个新的 Discovery 镜像。如果您已经引导主机，请使用最新的发现镜像再次引导它们。

1. 运行以下命令来应用 YAML 内容：

```
oc apply -f nmstateconfig.yaml
```

1.5.3.5.3. 其他资源

- 请参阅 [使用命令行界面创建主机清单](#)
- 请参阅 [Declarative Network API](#)

1.5.3.6. 使用 Discovery 镜像将主机添加到主机清单中

创建主机清单（基础架构环境）后，您可以发现主机并将其添加到清单中。要将主机添加到清单，请选择下载 ISO 并将其附加到每台服务器的方法。例如，您可以使用虚拟介质下载 ISO，或者将 ISO 写入 USB 驱动器。

重要：要防止安装失败，请在安装过程中保留发现 ISO 介质连接到该设备，并将每个主机设置为从设备引导。

1.5.3.6.1. 先决条件

- 您必须启用中央基础架构管理服务。如需更多信息，请参阅 [启用中央基础架构管理服务](#)。

- 您必须创建主机清单。如需更多信息，[请参阅使用控制台创建主机清单](#)。

1.5.3.6.2. 使用控制台添加主机

通过完成以下步骤下载 ISO：

1. 在控制台中选择 **Infrastructure > Host inventory**。
2. 从列表中选择您的基础架构环境。
3. 单击 **Add hosts** 并选择 **With Discovery ISO**。

现在，您会看到下载 ISO 的 URL。引导的主机会出现在主机清单表中。显示主机可能需要几分钟时间。您必须批准每个主机，然后才能使用它。您可以通过单击 **Actions** 并选择 **Approve** 来从清单表中选择主机。

1.5.3.6.3. 使用命令行界面添加主机

您可以在 **InfraEnv** 资源状态中的 **isoDownloadURL** 属性中看到下载 ISO 的 URL。如需有关 **InfraEnv** 资源的更多信息，[请参阅使用命令行界面创建主机清单](#)。

每个引导的主机在同一命名空间中创建一个 **Agent** 资源。您必须批准每个主机，然后才能使用它。

1.5.3.6.4. 其他资源

- [请参阅启用中央基础架构管理服务](#)
- [请参阅使用控制台创建主机清单](#)
- [请参阅使用命令行界面创建主机清单](#)

1.5.3.7. 自动将裸机主机添加到主机清单中

创建主机清单（基础架构环境）后，您可以发现主机并将其添加到清单中。您可以通过为每个主机创建 **BareMetalHost** 资源和相关 BMC secret，使裸机 Operator 与每个裸机主机的 Baseboard Management Controller (BMC) 通信来自动引导基础架构环境的发现镜像。自动化由 **BareMetalHost** 上的标签设置，该标签引用您的基础架构环境。

自动化执行以下操作：

- 使用基础架构环境代表的发现镜像引导每个裸机主机
- 在更新基础架构环境或任何关联的网络配置时，使用最新发现镜像重启每个主机
- 在发现时将每个 **Agent** 资源与对应的 **BareMetalHost** 资源关联
- 根据 **BareMetalHost** 的信息更新 **Agent** 资源属性，如主机名、角色和安装磁盘
- 批准将代理用作集群节点

1.5.3.7.1. 先决条件

- 您必须启用中央基础架构管理服务。
- 您必须创建主机清单。

1.5.3.7.2. 使用控制台添加裸机主机

完成以下步骤，使用控制台自动将裸机主机添加到主机清单中：

1. 在控制台中选择 **Infrastructure > Host inventory**。
2. 从列表中选择您的基础架构环境。
3. 点 **Add hosts** 并选择 **With BMC Form**。
4. 添加所需信息并点 **Create**。

1.5.3.7.3. 使用命令行界面添加裸机主机

完成以下步骤，使用命令行界面自动将裸机主机添加到主机清单中。

1. 通过应用以下 YAML 内容并在需要时替换值来创建 BMC secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: <bmc-secret-name>
  namespace: <your_infraenv_namespace> ❶
type: Opaque
data:
  username: <username>
  password: <password>
```

- ❶ 命名空间必须与 **InfraEnv** 的命名空间相同。

2. 通过应用以下 YAML 内容并替换所需的值来创建裸机主机：

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: <bmh-name>
  namespace: <your_infraenv_namespace> ❶
annotations:
  inspect.metal3.io: disabled
labels:
  infraenvs.agent-install.openshift.io: <your-infraenv> ❷
spec:
  online: true
  automatedCleaningMode: disabled ❸
  bootMACAddress: <your-mac-address> ❹
  bmc:
    address: <machine-address> ❺
    credentialsName: <bmc-secret-name> ❻
  rootDeviceHints:
    deviceName: /dev/sda ❼
```

- ❶ 命名空间必须与 **InfraEnv** 的命名空间相同。
- ❷ 名称必须与 **InfrEnv** 的名称匹配，并存在于同一命名空间中。

- 3 如果没有设置值，则会自动使用 **metadata** 值。
- 4 确保 MAC 地址与主机上一个交集的 MAC 地址匹配。
- 5 使用 BMC 的地址。如需更多信息，请参阅[带外管理 IP 地址的端口访问](#)。
- 6 确保 **credentialsName** 值与您创建的 BMC secret 的名称匹配。
- 7 可选：选择安装磁盘。有关可用根设备提示，请参阅 *BareMetalHost spec*。使用 Discovery 镜像引导主机并创建对应的 **Agent** 资源后，会根据提示设置安装磁盘。

打开主机后，镜像将开始下载。这可能需要几分钟时间。当发现主机时，会自动创建 **Agent** 自定义资源。

1.5.3.7.4. 其他资源

- 如需有关零接触置备的更多信息，请参阅 OpenShift Container Platform 文档中的[网络边缘集群](#)。
- 要了解使用裸机主机所需的端口，请参阅 OpenShift Container Platform 文档中的[带外管理 IP 地址的端口](#)。
- 要了解 root 设备提示，请参阅 OpenShift Container Platform 文档中的[BareMetalHost spec](#)。
- [请参阅使用镜像 pull secret](#)
- [请参阅为内部环境创建凭证](#)

1.5.3.8. 管理主机清单

您可以使用控制台管理主机清单并编辑现有主机，或使用命令行界面并编辑 **Agent** 资源。

1.5.3.8.1. 使用控制台管理主机清单

使用 Discovery ISO 成功引导的每个主机都显示为主机清单中的一行。您可以使用控制台编辑和管理主机。如果手动引导主机且没有使用裸机 Operator 自动化，则必须在控制台中批准主机，然后才能使用它。准备好作为 OpenShift 节点安装的主机具有 **Available** 状态。

1.5.3.8.2. 使用命令行界面管理主机清单

Agent 资源代表每个主机。您可以在 **Agent** 资源中设置以下属性：

- **clusterDeploymentName**
如果要将主机安装为集群中的节点，请将此属性设置为要使用的 **ClusterDeployment** 的命名空间和名称。
- 可选：**role**
设置集群中主机的角色。可能的值有 **master**、**worker** 和 **auto-assign**。默认值为 **auto-assign**。
- **hostname**
设置主机的主机名。可选：如果主机被自动分配一个有效的主机名，例如使用 DHCP。
- 已批准

此属性如果未设置，则默认为空。如果主机被自动分配一个有效的主机名，则此属性默认为空。如果主机

指明主机是否可以作为 OpenShift 节点安装。此属性是一个布尔值，默认值为 **False**。如果手动引导主机且没有使用裸机 Operator 自动化，则必须在安装主机前将此属性设置为 **True**。

- **installation_disk_id**

您所选的安装磁盘的 ID 在主机的清单中可见。

- **installerArgs**

包含主机的 *coreos-installer* 参数覆盖的 JSON 格式字符串。您可以使用此属性修改内核参数。请参见以下示例语法：

```
[ "--append-karg",
  "ip=192.0.2.2::192.0.2.254:255.255.255.0:core0.example.com:enp1s0:none", "--save-partindex", "4" ]
```

- **ignitionConfigOverrides**

包含主机的 ignition 配置覆盖的 JSON 格式字符串。您可以使用 ignition 将文件添加到主机中。请参见以下示例语法：

```
{ "ignition": { "version": "3.1.0" }, "storage": { "files": [ { "path": "/tmp/example", "contents": { "source": "data:text/plain;base64,aGVscGlt dHJhcHBIZGluYXN3YWdnZXJzcGVj" } } ] } }
```

- **nodeLabels**

安装主机后应用到节点的标签列表。Agent 资源的状态具有以下属性：

- **role**

设置集群中主机的角色。如果您之前在 **Agent** 资源中设置了角色，则该值会出现在状态中。

- **清单 (inventory)**

包含在主机发现上运行的代理的主机属性。

- **progress**

主机安装进度。

- **ntpSources**

主机配置的网络时间协议(NTP)源。

-

conditions

包含以下标准 **Kubernetes** 条件，带有 **True** 或 **False** 值：

-

SpecSynced：如果所有指定属性都成功应用，则为 **True**。如果遇到一些错误，则为 **false**。

-

connected：如果代理连接到安装服务，则为 **True**。如果代理在一段时间内没有联系安装服务，则为 **false**。

-

requirementsMet: 如果主机准备好开始安装，则为 **True**。

-

validated: 如果所有主机验证都通过，则为 **True**。

-

installed：如果主机作为 **OpenShift** 节点安装，则为 **True**。

-

bound: 如果主机绑定到集群，则为 **True**。

-

cleanup：如果删除 **Agent resource** 的请求失败，则为 **False**。

-

debugInfo

包含用于下载安装日志和事件的 **URL**。

-

validationsInfo

包含有关代理在发现主机后运行的验证信息，以确保安装成功。如果值为 **False**，则进行故障排除。

-

installation_disk_id

您所选的安装磁盘的 ID 在主机的清单中可见。

1.5.3.8.3. 其他资源

- 请参阅 [访问主机清单](#)
- 请参阅 [coreos-installer install](#)

1.5.4. 创建集群

了解如何使用多集群引擎 **operator** 在云供应商中创建 **Red Hat OpenShift Container Platform** 集群。

multicluster engine operator 使用 **OpenShift Container Platform** 提供的 **Hive operator** 为除内部集群和托管 **control plane** 以外的所有供应商置备集群。在置备内部集群时，多集群引擎 **operator** 使用 **OpenShift Container Platform** 提供的中央基础架构管理和辅助安装程序功能。托管 **control plane** 托管的集群使用 **HyperShift operator** 置备。

- [在集群创建过程中配置额外清单](#)
- [在 Amazon Web Services 上创建集群](#)
- [在 Amazon Web Services GovCloud 上创建集群](#)
- [在 Microsoft Azure 上创建集群](#)
- [在 Google Cloud Platform 上创建集群](#)
- [在 VMware vSphere 上创建集群](#)
- [在 Red Hat OpenStack Platform 上创建集群](#)

- [在 Red Hat Virtualization 上创建集群](#)
- [在内部环境中创建集群](#)
- [托管 control plane（技术预览）](#)

1.5.4.1. 使用 CLI 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。

- [先决条件](#)
- [使用 ClusterDeployment 创建集群](#)
- [使用集群池创建集群](#)

1.5.4.1.1. 先决条件

在创建集群前，您必须克隆 **clusterImageSets** 存储库，并将其应用到 **hub** 集群。请参见以下步骤：

1. 运行以下命令来克隆：

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/backplane-2.2
```

2. 运行以下命令将其应用到 **hub** 集群：

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

在创建集群时选择 **Red Hat OpenShift Container Platform** 发行镜像。

1.5.4.1.2. 使用 `ClusterDeployment` 创建集群

`ClusterDeployment` 是一个 `Hive` 自定义资源，用于控制集群的生命周期。

按照 [使用 `Hive`](#) 文档创建 `ClusterDeployment` 自定义资源并创建单个集群。

1.5.4.1.3. 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。

按照 [Cluster Pools](#) 文档，使用 `Hive ClusterPool API` 创建集群。

1.5.4.2. 在集群创建过程中配置额外清单

您可以在创建集群的过程中配置额外的 `Kubernetes` 资源清单。如果您需要为配置网络或设置负载均衡器等场景配置额外清单，这可以提供帮助。

在创建集群前，您需要添加对 `ClusterDeployment` 资源的引用，该资源指定包含其他资源清单的 `ConfigMap`。

注：`ClusterDeployment` 资源和 `ConfigMap` 必须位于同一命名空间中。以下示例演示了您的内容看起来如何。

- 带有资源清单的 `ConfigMap`

包含具有另一个 `ConfigMap` 资源的清单的 `ConfigMap`。资源清单 `ConfigMap` 可以包含多个键，并在 `data.<resource_name>\.yaml` 模式中添加资源配置。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: <my-baremetal-cluster-install-manifests>
  namespace: <mynamespace>
data:
  99_metal3-config.yaml: |
    kind: ConfigMap
    apiVersion: v1
    metadata:
      name: metal3-config
```

```

namespace: openshift-machine-api
data:
  http_port: "6180"
  provisioning_interface: "enp1s0"
  provisioning_ip: "172.00.0.3/24"
  dhcp_range: "172.00.0.10,172.00.0.100"
  deploy_kernel_url: "http://172.00.0.3:6180/images/ironic-python-agent.kernel"
  deploy_ramdisk_url: "http://172.00.0.3:6180/images/ironic-python-agent.initramfs"
  ironic_endpoint: "http://172.00.0.3:6385/v1/"
  ironic_inspector_endpoint: "http://172.00.0.3:5150/v1/"
  cache_url: "http://192.168.111.1/images"
  rhcos_image_url: "https://releases-art-
rhcos.svc.ci.openshift.org/art/storage/releases/rhcos-
4.3/43.81.201911192044.0/x86_64/rhcos-43.81.201911192044.0-
openstack.x86_64.qcow2.gz"

```

- 使用引用的资源清单 **ConfigMap** 的 **ClusterDeployment**

资源清单 **ConfigMap** 在 **spec.provisioning.manifestsConfigMapRef** 下引用。

```

apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: <my-baremetal-cluster>
  namespace: <mynamespace>
  annotations:
    hive.openshift.io/try-install-once: "true"
spec:
  baseDomain: test.example.com
  clusterName: <my-baremetal-cluster>
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    baremetal:
      libvirtSSHPrivateKeySecretRef:
        name: provisioning-host-ssh-private-key
  provisioning:
    installConfigSecretRef:
      name: <my-baremetal-cluster-install-config>
    sshPrivateKeySecretRef:
      name: <my-baremetal-hosts-ssh-private-key>
    manifestsConfigMapRef:
      name: <my-baremetal-cluster-install-manifests>
    imageSetRef:
      name: <my-clusterimageset>
    sshKnownHosts:
      - "10.1.8.90 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXvVVVKUYVkuYvkuYgkuyTCYTytfkufTYAAAAIbmlzdHAyNTYAAABB
BKWjJRzeUVuZs4yxSy4eu45xiANFIbwE3e1aPzGD58x/NX7Yf+S8eFKq4RrsfSaK2hVJyJ
jvVlhUsU9z2sBJP8="
    pullSecretRef:
      name: <my-baremetal-cluster-pull-secret>

```

1.5.4.3. 在 Amazon Web Services 上创建集群

您可以使用 **multicluster engine operator** 控制台在 **Amazon Web Services (AWS)** 上创建 **Red Hat OpenShift Container Platform** 集群。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 [AWS 上安装](#) 有关流程的更多信息。

- [先决条件](#)
- [创建 AWS 集群](#)
- [使用控制台创建集群](#)

1.5.4.3.1. 先决条件

在 **AWS** 上创建集群前请查看以下先决条件：

- 您必须已部署 **hub** 集群。
- 您需要 **AWS** 凭证。如需更多信息，请参阅 [Amazon Web Services 创建凭证](#)。
- 您在 **AWS** 中需要配置的域。有关如何配置域的说明，请参阅 [配置 AWS 帐户](#)。
- 您需要具有 **Amazon Web Services (AWS)** 登录凭证，其中包括用户名、密码、访问密钥 ID 和 **secret** 访问密钥。请参阅 [了解和获取您的安全凭证](#)。
- 您必须具有 **OpenShift Container Platform** 镜像 **pull secret**。请参阅 [使用镜像 pull secret](#)。

注：如果您在云供应商上更改云供应商访问密钥，您还需要在控制台中手动更新云供应商的对应凭证。当凭证在托管受管集群的云供应商过期并尝试删除受管集群时，需要此项。

1.5.4.3.2. 创建 AWS 集群

请参阅以下有关创建 **AWS** 集群的重要信息：

- 当您在创建集群前查看信息并选择性地自定义它时，您可以选择 **YAML: On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。
- 当您创建集群时，控制器会为集群和资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。
- **销毁集群** 会删除命名空间和所有资源。
- 如果要添加集群到现有的集群集中，则需要为集群设置具有正确权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。
- 如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。
- 每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。
- 如果已有与您使用 **AWS** 帐户配置的所选凭证关联的基本 **DNS** 域，则该值会在字段中填充。您可以修改它的值来覆盖它。此名称用于集群的主机名。
- 发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。从可用的镜像列表中选择镜像。如果要使用的镜像不可用，您可以输入您要使用的镜像的 **URL**。
- 节点池包括 **control plane** 池和 **worker** 池。**control plane** 节点共享集群活动的管理。该信息包括以下字段：
 - **region**：指定您要节点池的区域。
 - **CPU 架构**：如果受管集群的架构类型与 **hub** 集群的架构不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**, **ppc64le**, **s390x**, 和 **arm64**。

- **Zones** : 指定您要运行 **control plane** 池的位置。您可以为更分布式的 **control plane** 节点选择区域中的多个区。距离更近的区域可能会提供更快的性能, 但距离更远的区域可能更为分散。
- **实例类型** : 指定 **control plane** 节点的实例类型。您可以在实例创建后更改实例的类型和大小。
- **根存储** : 指定要为集群分配的 **root** 存储量。
- 您可以在 **worker** 池中创建零个或多个 **worker** 节点, 以运行集群的容器工作负载。这可以位于单个 **worker** 池中, 也可以分布到多个 **worker** 池中。如果指定了零个 **worker** 节点, **control plane** 节点也充当 **worker** 节点。可选信息包括以下字段:
 - **Zones** : 指定您要运行 **worker** 池的位置。您可以为更加分散的节点组群选择区域中的多个区。距离更近的区域可能会提供更快的性能, 但距离更远的区域可能更为分散。
 - **实例类型** : 指定 **worker** 池的实例类型。您可以在实例创建后更改实例的类型和大小。
 - **节点数** : 指定 **worker** 池的节点计数。定义 **worker** 池时需要此设置。
 - **根存储** : 指定分配给 **worker** 池的根存储量。定义 **worker** 池时需要此设置。
- 集群需要网络详情, 并使用 IPv6 需要多个网络。您可以通过点 **Add network** 来添加额外网络。
- 凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样, 覆盖这些信息, 或者在要启用代理时添加信息。以下列表包含创建代理所需的信息:
 - **HTTP proxy** : 指定应用作 HTTP 流量的代理的 URL。
 - **HTTPS 代理** : 指定用于 HTTPS 流量的安全代理 URL。如果没有提供值, 则使用相同的值 HTTP Proxy URL, 用于 HTTP 和 HTTPS。

- 没有代理站点：应绕过代理的、以逗号分隔的站点列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。
- 其他信任捆绑包：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

1.5.4.3.3. 使用控制台创建集群

要创建新集群，请参阅以下步骤。如果您希望导入一个现有的集群，请参阅[将目标受管集群导入到 hub 集群](#)。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine operator** 管理自动配置。

1. 进入 **Infrastructure > Clusters**。
2. 在 **Clusters** 页面上。点 **Cluster > Create cluster** 并完成控制台中的步骤。
3. 可选：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果您需要创建凭证，请参阅[为 Amazon Web Services 创建凭证](#)。

集群的主机名用于集群的主机名。

1.5.4.3.4. 其他资源

- 在创建 **AWS GovCloud** 集群时使用 **AWS** 私有配置信息。有关 [在该环境中创建集群的信息](#)，请参阅在 [Amazon Web Services GovCloud](#) 上创建集群。
- 如需更多信息，请参阅[配置 AWS 帐户](#)。

- 有关 [发行镜像的更多信息](#)，请参阅[指定发行镜像](#)。
- 通过访问您的云供应商站点（如 [AWS 通用实例](#)）[查找有关支持的即时类型的更多信息](#)。

1.5.4.4. 在 Amazon Web Services GovCloud 上创建集群

您可以使用控制台在 **Amazon Web Services (AWS)** 或 **AWS GovCloud** 上创建 **Red Hat OpenShift Container Platform** 集群。此流程解释了如何在 **AWS GovCloud** 上创建集群。有关在 **AWS** 上创建集群的说明，请参阅在 [Amazon Web Services](#) 上创建集群。

AWS GovCloud 提供满足在云上存储政府文档的额外要求的云服务。在 **AWS GovCloud** 上创建集群时，您必须完成额外的步骤来准备您的环境。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 [AWS 上安装集群到政府区域](#)，以了解有关流程的更多信息。以下小节提供了在 **AWS GovCloud** 上创建集群的步骤：

- [先决条件](#)
- [配置 Hive 以在 AWS GovCloud 上部署](#)
- [使用控制台创建集群](#)

1.5.4.4.1. 先决条件

创建 **AWS GovCloud** 集群前必须满足以下先决条件：

- 您必须具有 **AWS** 登录凭证，其中包括用户名、密码、访问密钥 ID 和 **secret** 访问密钥。请参阅[了解和获取您的安全凭证](#)。
- 您需要 **AWS** 凭证。如需更多信息，请参阅为 [Amazon Web Services](#) 创建凭证。

- 您在 **AWS** 中需要配置的域。有关如何配置域的说明，[请参阅配置 AWS 帐户](#)。
- 您必须具有 **OpenShift Container Platform** 镜像 **pull secret**。请参阅[使用镜像 pull secret](#)。
- 您必须有一个带有 **hub** 集群的现有 **Red Hat OpenShift Container Platform** 集群的 **Amazon Virtual Private Cloud (VPC)**。此 **VPC** 必须与用于受管集群资源或受管集群服务端点的 **VPC** 不同。
- 您需要部署受管集群资源的 **VPC**。这不能与用于 **hub** 集群或受管集群服务端点的 **VPC** 相同。
- 您需要一个或多个提供受管集群服务端点的 **VPC**。这不能与用于 **hub** 集群或受管集群的 **VPC** 相同。
- 确保由无类别域间路由(**CIDR**)指定的 **VPC IP** 地址不会重叠。
- 您需要一个 **HiveConfig** 自定义资源来引用 **Hive** 命名空间中的凭证。此自定义资源必须有权访问您为受管集群服务端点创建的 **VPC** 上创建资源。

注：如果您在云供应商上更改云供应商访问密钥，您还需要在 **multicluster engine operator** 控制台中手动更新云供应商的对应凭证。当凭证在托管受管集群的云供应商过期并尝试删除受管集群时，需要此项。

1.5.4.4.2. 配置 **Hive** 以在 **AWS GovCloud** 上部署

在 **AWS GovCloud** 上创建集群时与在标准 **AWS** 上创建集群时，您必须完成一些额外的步骤，以便在 **AWS GovCloud** 上为集群准备 **AWS PrivateLink**。

1.5.4.4.2.1. 为资源和端点创建 **VPC**

如先决条件中所示，除了包含 **hub** 集群的 **VPC** 外，还需要两个 **VPC**。有关创建 **VPC** 的具体步骤，请参阅 **Amazon Web Services** 文档中的创建 **VPC**。

1. 使用专用子网为受管集群创建 **VPC**。
2. 使用专用子网为受管集群服务端点创建一个或多个 **VPC**。区域中的每个 **VPC** 限制为 **255** 个 **VPC** 端点，因此您需要多个 **VPC** 来支持该区域中的 **255** 个集群。
3. 对于每个 **VPC**，在区域的所有支持的可用区中创建子网。由于控制器要求，每个子网必须至少有 **255** 个可用 **IP** 地址。

以下示例演示了如何为在 **us-gov-east-1** 区域中有 **6** 个可用区的 **VPC** 来构建子网：

```
vpc-1 (us-gov-east-1) : 10.0.0.0/20
subnet-11 (us-gov-east-1a): 10.0.0.0/23
subnet-12 (us-gov-east-1b): 10.0.2.0/23
subnet-13 (us-gov-east-1c): 10.0.4.0/23
subnet-12 (us-gov-east-1d): 10.0.8.0/23
subnet-12 (us-gov-east-1e): 10.0.10.0/23
subnet-12 (us-gov-east-1f): 10.0.12.0/2
```

```
vpc-2 (us-gov-east-1) : 10.0.16.0/20
subnet-21 (us-gov-east-1a): 10.0.16.0/23
subnet-22 (us-gov-east-1b): 10.0.18.0/23
subnet-23 (us-gov-east-1c): 10.0.20.0/23
subnet-24 (us-gov-east-1d): 10.0.22.0/23
subnet-25 (us-gov-east-1e): 10.0.24.0/23
subnet-26 (us-gov-east-1f): 10.0.28.0/23
```

4. 确保所有 **hub** 环境(**hub** 集群 **VPC**)都有与您为使用对等、传输网关和所有 **DNS** 设置创建的 **VPC** 创建的网络连接。
5. 收集解决 **AWS PrivateLink** 的 **DNS** 设置所需的 **VPC** 列表，这是 **AWS GovCloud** 连接所需的。这包括您要配置的 **multicloud engine operator** 实例的 **VPC**，并可以包含所有存在 **Hive** 控制器的 **VPC** 列表。

1.5.4.4.2.2. 为 **VPC** 端点配置安全组

AWS 中的每个 **VPC** 端点都附加了一个安全组来控制对端点的访问。当 **Hive** 创建 **VPC** 端点时，它没有指定安全组。**VPC** 的默认安全组附加到 **VPC** 端点。**VPC** 的默认安全组必须具有规则，以允许从 **Hive** 安装程序 **pod** 创建 **VPC** 端点的流量。详情请参阅 [AWS 文档中的使用端点策略控制对 **VPC** 端点的访问](#)。

例如，如果 **Hive** 在 **hive-vpc (10.1.0.0/16)** 中运行，则 **VPC** 端点在创建 **VPC** 端点的默认安全组中

必须有一个规则，它允许从 **10.1.0.0/16** 进行入口。

1.5.4.4.2.3. 为 **AWS PrivateLink** 设置权限

您需要多个凭证来配置 **AWS PrivateLink**。这些凭证所需的权限取决于凭证类型。

- **ClusterDeployment** 的凭证需要以下权限：

```
ec2:CreateVpcEndpointServiceConfiguration
ec2:DescribeVpcEndpointServiceConfigurations
ec2:ModifyVpcEndpointServiceConfiguration
ec2:DescribeVpcEndpointServicePermissions
ec2:ModifyVpcEndpointServicePermissions
ec2>DeleteVpcEndpointServiceConfigurations
```

- 端点 **VPCs** 帐户 `.spec.awsPrivateLink.credentialsSecretRef` 的 **HiveConfig** 凭证需要以下权限：

```
ec2:DescribeVpcEndpointServices
ec2:DescribeVpcEndpoints
ec2:CreateVpcEndpoint
ec2:CreateTags
ec2:DescribeNetworkInterfaces
ec2:DescribeVPCs

ec2>DeleteVpcEndpoints

route53:CreateHostedZone
route53:GetHostedZone
route53:ListHostedZonesByVPC
route53:AssociateVPCWithHostedZone
route53:DisassociateVPCFromHostedZone
route53:CreateVPCAssociationAuthorization
route53>DeleteVPCAssociationAuthorization
route53:ListResourceRecordSets
route53:ChangeResourceRecordSets

route53>DeleteHostedZone
```

- **HiveConfig** 自定义资源中指定的凭证将 **VPC** 与私有托管区 (`.spec.awsPrivateLink.associatedVPCs[$idx].credentialsSecretRef`) 关联。VPC 所在的帐户需要以下权限：

```
route53:AssociateVPCWithHostedZone
route53:DisassociateVPCFromHostedZone
ec2:DescribeVPCs
```

确保 **hub** 集群上的 **Hive** 命名空间中有一个凭证 **secret**。

HiveConfig 自定义资源需要引用 **Hive** 命名空间中的凭证，该凭证具有在特定提供的 **VPC** 中创建资源的权限。如果要在 **AWS GovCloud** 中置备 **AWS** 集群的凭证已在 **Hive** 命名空间中，则不需要创建另一个凭证。如果您要在 **AWS GovCloud** 中置备 **AWS** 集群的凭证还没有在 **Hive** 命名空间中，您可以替换当前凭证或在 **Hive** 命名空间中创建额外凭证。

HiveConfig 自定义资源需要包含以下内容：

- 具有为给定 **VPC** 置备资源所需的权限的 **AWS GovCloud** 凭证。
- **OpenShift Container Platform** 集群安装的 **VPC** 地址以及受管集群的服务端点。

最佳实践：为 **OpenShift Container Platform** 集群安装和服务端点使用不同的 **VPC**。

以下示例显示了凭证内容：

```
spec:
  awsPrivateLink:
    ## The list of inventory of VPCs that can be used to create VPC
    ## endpoints by the controller.
    endpointVPCInventory:
      - region: us-east-1
        vpcID: vpc-1
        subnets:
          - availabilityZone: us-east-1a
            subnetID: subnet-11
          - availabilityZone: us-east-1b
            subnetID: subnet-12
          - availabilityZone: us-east-1c
            subnetID: subnet-13
          - availabilityZone: us-east-1d
            subnetID: subnet-14
          - availabilityZone: us-east-1e
            subnetID: subnet-15
          - availabilityZone: us-east-1f
            subnetID: subnet-16
      - region: us-east-1
        vpcID: vpc-2
        subnets:
          - availabilityZone: us-east-1a
            subnetID: subnet-21
          - availabilityZone: us-east-1b
```

```

    subnetID: subnet-22
  - availabilityZone: us-east-1c
    subnetID: subnet-23
  - availabilityZone: us-east-1d
    subnetID: subnet-24
  - availabilityZone: us-east-1e
    subnetID: subnet-25
  - availabilityZone: us-east-1f
    subnetID: subnet-26
## The credentialsSecretRef points to a secret with permissions to create.
## The resources in the account where the inventory of VPCs exist.
credentialsSecretRef:
  name: <hub-account-credentials-secret-name>

## A list of VPC where various mce clusters exists.
associatedVPCs:
- region: region-mce1
  vpcID: vpc-mce1
  credentialsSecretRef:
    name: <credentials-that-have-access-to-account-where-MCE1-VPC-exists>
- region: region-mce2
  vpcID: vpc-mce2
  credentialsSecretRef:
    name: <credentials-that-have-access-to-account-where-MCE2-VPC-exists>

```

您可以从 端点 `VPCInventory` 列表中支持 `AWS PrivateLink` 的所有区域包含一个 `VPC`。控制器选择一个满足 `ClusterDeployment` 要求的 `VPC`。

如需更多信息，请参阅 [Hive 文档](#)。

1.5.4.4.3. 使用控制台创建集群

要从控制台创建集群，请导航到 `Infrastructure > Clusters > Create cluster AWS > Standalone`，并完成控制台中的步骤。

注：此过程用于创建集群。如果您有一个要导入的现有集群，请参阅 [将目标受管集群导入到 hub 集群](#) 以了解这些步骤。

如果创建一个 `AWS GovCloud` 集群，则您选择的凭证必须有权访问 `AWS GovCloud` 区域中的资源。如果有部署集群所需的权限，您可以使用 `Hive` 命名空间中已存在的 `AWS GovCloud secret`。在控制台中显示现有凭证。如果您需要创建凭证，请参阅 [为 Amazon Web Services 创建凭证](#)。

集群的主机名用于集群的主机名。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要添加集群到现有的集群集中，则需要为集群设置具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与您使用 **AWS** 或 **AWS GovCloud** 帐户配置的所选凭证关联的基本 **DNS** 域，则该值会在字段中填充。您可以修改它的值来覆盖它。此名称用于集群的主机名。如需更多信息，[请参阅配置 AWS 帐户](#)。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 **URL**。有关 [发行镜像的更多信息](#)，[请参阅指定发行镜像](#)。

节点池包括 **control plane** 池和 **worker** 池。**control plane** 节点共享集群活动的管理。该信息包括以下字段：

- **region**：创建集群资源的区域。如果要在 **AWS GovCloud** 供应商上创建集群，则必须为节点池包含 **AWS GovCloud** 区域。例如，**us-gov-west-1**。
- **CPU 架构**：如果受管集群的架构类型与 **hub** 集群的架构不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**, **ppc64le**, **s390x**, 和 **arm64**。
- **Zones**：指定您要运行 **control plane** 池的位置。您可以为更分布式的 **control plane** 节点选择区域中的多个区。距离更近的区域可能会提供更高的性能，但距离更远的区域可能更为分散。
- **实例类型**：指定 **control plane** 节点的实例类型，它必须与之前所示的 **CPU 架构** 相同。您可以在实例创建后更改实例的类型和大小。

- 根存储：指定要为集群分配的 **root** 存储量。

您可以在 **worker** 池中创建零个或多个 **worker** 节点，以运行集群的容器工作负载。它们可以位于单个 **worker** 池中，也可以分布在多个 **worker** 池中。如果指定了零个 **worker** 节点，**control plane** 节点也充当 **worker** 节点。可选信息包括以下字段：

- 池名称：为您的池提供唯一名称。
- **Zones**：指定您要运行 **worker** 池的位置。您可以为更加分散的节点组群选择区域中的多个区。距离更近的区域可能会提供更快的性能，但距离更远的区域可能更为分散。
- 实例类型：指定 **worker** 池的实例类型。您可以在实例创建后更改实例的类型和大小。
- 节点数：指定 **worker** 池的节点计数。定义 **worker** 池时需要此设置。
- 根存储：指定分配给 **worker** 池的根存储量。定义 **worker** 池时需要此设置。

集群需要网络详情，并使用 **IPv6** 需要多个网络。对于 **AWS GovCloud** 集群，在 **Machine CIDR** 字段中输入 **Hive VPC** 地址块的值。您可以通过点 **Add network** 来添加额外网络。

凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样，覆盖这些信息，或者在要启用代理时添加信息。以下列表包含创建代理所需的信息：

- **HTTP 代理 URL**：指定应当用作 **HTTP** 流量的代理的 **URL**。
- **HTTPS 代理 URL**：指定用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- 无代理域：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。

- 其他信任捆绑包：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

在创建 **AWS GovCloud** 集群或使用私有环境时，请使用 **AMI ID** 和子网值完成 **AWS 私有配置页面** 中的字段。在 **ClusterDeployment.yaml** 文件中，确保 **spec:platform:aws:privateLink:enabled** 的值被设置为 **true**，该文件在选择 **Use private configuration** 时会自动设置。

当您在创建集群前查看信息并选择性地自定义它时，您可以选择 **YAML: On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine for Kubernetes operator** 管理自动配置。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.5. 在 Microsoft Azure 上创建集群

您可以使用 **multicluster engine operator** 控制台在 **Microsoft Azure** 或 **Microsoft Azure Government** 上部署 **Red Hat OpenShift Container Platform** 集群。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 [Azure 上安装](#) 有关流程的更多信息。

- [先决条件](#)
- [使用控制台创建集群](#)

1.5.4.5.1. 先决条件

在 **Azure** 上创建集群前请查看以下先决条件：

- 您必须已部署 **hub** 集群。

- 您需要一个 **Azure** 凭证。如需更多信息，请参阅为 [Microsoft Azure 创建凭证](#)。
- 您需要在 **Azure** 或 **Azure Government** 中配置了域。有关如何配置域的说明，请参阅为 [Azure 云服务配置自定义域名](#)。
- 您需要 **Azure** 登录凭证，其中包括用户名和密码。请参阅 [Microsoft Azure Portal](#)。
- 您需要 **Azure** 服务主体，其中包括 `clientId`、`clientSecret` 和 `tenantId`。请参阅 [azure.microsoft.com](#)。
- 您需要 **OpenShift Container Platform** 镜像 `pull secret`。请参阅[使用镜像 pull secret](#)。

注：如果您在云供应商上更改云供应商访问密钥，则需要在 **multicluster engine operator** 的控制台中手动更新云供应商的对应凭证。当凭证在托管受管集群的云供应商过期并尝试删除受管集群时，需要此项。

1.5.4.5.2. 使用控制台创建集群

要从 **multicluster engine operator** 控制台创建集群，请进入到 **Infrastructure > Clusters**。在 **Clusters** 页面上，点 **Create cluster** 并完成控制台中的步骤。

注：此过程用于创建集群。如果您有一个要导入的现有集群，请参阅[将目标受管集群导入到 hub 集群](#)以了解这些步骤。

如果您需要创建凭证，请参阅为 [Microsoft Azure 创建凭证](#)。

集群的主机名用于集群的主机名。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要添加集群到现有的集群集中，则需要为集群设置具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与您为 **Azure** 帐户配置的所选凭证关联的基本 **DNS** 域，则该值会在那个字段中填充。您可以修改它的值来覆盖它。如需更多信息，请参阅 [Azure 云服务配置自定义域名](#)。此名称用于集群的主机名。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 **URL**。 [发行镜像](#) 以了解有关发行镜像的更多信息。

Node 池包括 **control plane** 池和 **worker** 池。**control plane** 节点共享集群活动的管理。该信息包括以下可选字段：

- **region**：指定一个您要运行节点池的区域。您可以为更分布式的 **control plane** 节点选择区域中的多个区。距离更近的区域可能会提供更快的性能，但距离更远的区域可能更为分散。
- **CPU 架构**：如果受管集群的架构类型与 **hub** 集群的架构不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**, **ppc64le**, **s390x**, 和 **arm64**。

您可以在创建集群后更改 **control plane** 池的 **Instance type** 和 **Root** 存储分配（必需）。

您可以在 **worker** 池中创建一个或多个 **worker** 节点，以运行集群的容器工作负载。它们可以位于单个 **worker** 池中，也可以分布在多个 **worker** 池中。如果指定了零个 **worker** 节点，**control plane** 节点也充当 **worker** 节点。该信息包括以下字段：

- **Zones**：指定您要运行 **worker** 池的位置。您可以为更加分散的节点组群选择区域中的多个区。距离更近的区域可能会提供更快的性能，但距离更远的区域可能更为分散。
- **实例类型**：您可以在实例创建后更改实例的类型和大小。

您可以通过点 **Add network** 来添加额外网络。如果您使用的是 **IPv6** 地址，您必须有多个网络。

凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样，覆盖这些信息，或者在要启用代理时添加信息。以下列表包含创建代理所需的信息：

- **HTTP 代理**：用作 **HTTP** 流量的代理的 **URL**。
- **HTTPS 代理**：用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- **无代理**：应绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。
- **其他信任捆绑包**：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

当您在创建集群前查看信息并选择性地自定义它时，您可以点 **YAML 切换 On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine operator** 管理自动配置。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.6. 在 Google Cloud Platform 上创建集群

按照步骤在 **Google Cloud Platform (GCP)** 上创建 **Red Hat OpenShift Container Platform** 集群有关 **GCP** 的更多信息，请参阅 [Google Cloud Platform](#)。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 [GCP 上安装](#) 有关流程的更多信息。

- [先决条件](#)
- [使用控制台创建集群](#)

1.5.4.6.1. 先决条件

在 **GCP** 上创建集群前请查看以下先决条件：

- 您必须已部署 **hub** 集群。
- 您必须具有 **GCP** 凭证。如需更多信息，请参阅为 [Google Cloud Platform](#) 创建凭证。
- 您必须在 **GCP** 中配置了域。有关如何配置域的说明，请参阅[设置自定义域](#)。
- 您需要 **GCP** 登录凭证，其中包括用户名和密码。
- 您必须具有 **OpenShift Container Platform** 镜像 **pull secret**。请参阅[使用镜像 pull secret](#)。

注：如果您在云供应商上更改云供应商访问密钥，则需要在 **multicluster engine operator** 的控制台中手动更新云供应商的对应凭证。当凭证在托管受管集群的云供应商过期并尝试删除受管集群时，需要此项。

1.5.4.6.2. 使用控制台创建集群

要从 **multicluster engine operator** 控制台创建集群，请导航到 **Infrastructure > Clusters**。在 **Clusters** 页面上，点 **Create cluster** 并完成控制台中的步骤。

注：此过程用于创建集群。如果您有一个要导入的现有集群，请参阅[将目标受管集群导入到 hub 集群](#)以了解这些步骤。

如果您需要创建凭证，请参阅为 [Google Cloud Platform](#) 创建凭证。

集群名称用于集群的主机名。在命名 **GCP** 集群时有一些限制。这些限制包括，名称不要以 **goog** 开始；名称的任何部分都不要包含与 **google** 类似的内容。如需了解完整的限制列表，请参阅 [Bucket 命名指南](#)。

重要： 当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示： 在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要添加集群到现有的集群集中，则需要为集群设置具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与 **GCP** 帐户所选凭证关联的基本 **DNS** 域，则该值会在字段中填充。您可以修改它的值来覆盖它。如需更多信息，请参阅 [设置自定义域](#)。此名称用于集群的主机名。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 **URL**。 [发行镜像](#) 以了解有关发行镜像的更多信息。

Node 池包括 **control plane** 池和 **worker** 池。**control plane** 节点共享集群活动的管理。该信息包括以下字段：

- **Region**：指定您要运行 **control plane** 池的区域。距离更近的区域可能会提供更快性能，但距离更远的区域可能更为分散。
- **CPU 架构**：如果受管集群的架构类型与 **hub** 集群的架构不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**, **ppc64le**, **s390x**, 和 **arm64**。

您可以指定 **control plane** 池的实例类型。您可以在实例创建后更改实例的类型和大小。

您可以在 **worker** 池中创建一个或多个 **worker** 节点，以运行集群的容器工作负载。它们可以位于单个 **worker** 池中，也可以分布在多个 **worker** 池中。如果指定了零个 **worker** 节点，**control plane** 节点也充当 **worker** 节点。该信息包括以下字段：

- 实例类型：您可以在实例创建后更改实例的类型和大小。
- 节点数：定义 **worker** 池时需要此设置。

网络详细信息是必需的，需要使用 IPv6 地址的多个网络。您可以通过点 **Add network** 来添加额外网络。

凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样，覆盖这些信息，或者在要启用代理时添加信息。以下列表包含创建代理所需的信息：

- **HTTP 代理**：用作 **HTTP** 流量的代理的 **URL**。
- **HTTPS 代理**：用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- **没有代理站点**：应绕过代理的、以逗号分隔的站点列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。
- **其他信任捆绑包**：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

当您在创建集群前查看信息并选择性地自定义它时，您可以选择 **YAML: On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine operator** 管理自动配置。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.7. 在 VMware vSphere 上创建集群

您可以使用多集群引擎 **operator** 控制台在 **VMware vSphere** 上部署 **Red Hat OpenShift Container Platform** 集群。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 **vSphere 上安装** 有关流程的更多信息。

- [先决条件](#)
- [使用控制台创建集群](#)

1.5.4.7.1. 先决条件

在 **vSphere** 上创建集群前请查看以下先决条件：

- 您必须有一个在 **OpenShift Container Platform** 版本 **4.6** 或更高版本上部署的 **hub** 集群。
- 您需要 **vSphere** 凭证。如需更多信息，请参阅为 **VMware vSphere** 创建凭证。
- 您需要 **OpenShift Container Platform** 镜像 **pull secret**。请参阅[使用镜像 pull secret](#)。
- 您必须具有要部署的 **VMware** 实例的以下信息：
 - **API** 和 **Ingress** 实例所需的静态 **IP** 地址
 - 以下的 **DNS** 记录：
 - **api.<cluster_name>.<base_domain>**，它必须指向静态 **API VIP**
 - ***.apps.<cluster_name>.<base_domain>**，它必须指向 **Ingress VIP** 的静态 **IP**

地址

1.5.4.7.2. 使用控制台创建集群

要从 **multicluster engine operator** 控制台创建集群，请进入到 **Infrastructure > Clusters**。在 **Clusters** 页面上，点 **Create cluster** 并完成控制台中的步骤。

注：此过程用于创建集群。如果您有一个要导入的现有集群，请参阅[将目标受管集群导入到 hub 集群](#)以了解这些步骤。

如果您需要创建凭证，请参阅[VMware vSphere 创建凭证](#)，以了解有关创建凭证的更多信息。

集群名称用于集群的主机名。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要将集群添加到现有的集群集中，则需要在集群设置上具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与您为 **vSphere** 帐户配置的所选凭证关联的基域，则该值会在字段中填充。您可以修改它的值来覆盖它。如需更多信息，请参阅[使用自定义在 vSphere 上安装集群](#)。这个值必须与创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。此名称用于集群的主机名。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 **URL**。如需有关 [发行镜像的更多信息](#)，请参阅[指定发行镜像](#)

注：仅支持 **OpenShift Container Platform** 版本 **4.5.x** 或更高版本的发行镜像。

节点池包括 **control plane** 池和 **worker** 池。**control plane** 节点共享集群活动的管理。该信息包括 **CPU 架构** 字段。查看以下字段描述：

- **CPU 架构**：如果受管集群的架构类型与 **hub** 集群的架构不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**、**ppc64le**、**s390x**，和 **arm64**。

您可以在 **worker** 池中创建一个或多个 **worker** 节点，以运行集群的容器工作负载。它们可以位于单个 **worker** 池中，也可以分布在多个 **worker** 池中。如果指定了零个 **worker** 节点，**control plane** 节点也充当 **worker** 节点。该信息包括 **每个插槽的内核数**、**CPU**、**Memory_min MiB**、**_Disk** 大小（以 **GiB** 为单位）和 **节点数**。

需要网络信息。使用 **IPv6** 需要多个网络。一些所需网络信息包括以下字段：

- **vSphere 网络名**：指定 **VMware vSphere** 网络名称。
- **API VIP**：指定用于内部 **API** 通信的 **IP** 地址。

注：这个值必须与您用来创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。如果没有提供，**DNS** 必须预先配置，以便 **api.** 可以正确解析。

- **Ingress VIP**：指定用于入口流量的 **IP** 地址。

注：这个值必须与您用来创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。如果没有提供，则必须预先配置 **DNS**，以便 **test.apps** 可以被正确解析。

您可以通过点 **Add network** 来添加额外网络。如果您使用的是 **IPv6** 地址，您必须有多个网络。

凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样，覆盖这些信息，或者在要启用代理时添加信息。以下列表包含创建代理所需的信息：

- **HTTP proxy**：指定应用作 **HTTP** 流量的代理的 **URL**。

- **HTTPS 代理**：指定用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- **没有代理站点**：提供以逗号分隔的站点列表，这些站点应绕过代理。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。
- **其他信任捆绑包**：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

您可以点 **Disconnected** 安装来定义断开连接的安装镜像。当使用 **Red Hat OpenStack Platform** 供应商和断开连接的安装创建集群时，如果需要一个证书才能访问镜像 **registry**，在配置集群时需要在 **Configuration for disconnected installation** 段中的 **Additional trust bundle** 字段中输入它，在创建集群时在 **Disconnected installation** 段中输入它。

您可以点击 **Add Automation template** 来创建模板。

当您在创建集群前查看信息并选择性地自定义它时，您可以点 **YAML** 切换 **On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine operator** 管理自动配置。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.8. 在 Red Hat OpenStack Platform 上创建集群

您可以使用多集群引擎 **operator** 控制台在 **Red Hat OpenStack Platform** 上部署 **Red Hat OpenShift Container Platform** 集群。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 [OpenStack 上安装 OpenStack](#) 以了解有关流程的更多信息。

- [先决条件](#)

- [使用控制台创建集群](#)

1.5.4.8.1. 先决条件

在 **Red Hat OpenStack Platform** 上创建集群前请查看以下先决条件：

- 您必须有一个在 **OpenShift Container Platform** 版本 4.6 或更高版本上部署的 **hub** 集群。
- 您必须具有 **Red Hat OpenStack Platform** 凭证。如需更多信息，请参阅 [Red Hat OpenStack Platform 创建凭证](#)。
- 您需要 **OpenShift Container Platform** 镜像 **pull secret**。请参阅 [使用镜像 pull secret](#)。
- 您要部署的 **Red Hat OpenStack Platform** 实例需要以下信息：
 - **control plane** 和 **worker** 实例的 **flavor** 名称，如 **m1.xlarge**
 - 外部网络的名称，以提供浮动 **IP** 地址
 - **API** 和入口实例所需的浮动 **IP** 地址
 - 以下的 **DNS** 记录：
 - **api.<cluster_name>.<base_domain>**，它必须指向 **API** 的浮动 **IP** 地址
 - ***.apps.<cluster_name>.<base_domain>**，它必须指向 **ingress** 的浮动 **IP** 地址

1.5.4.8.2. 使用控制台创建集群

要从 **multicluster engine operator** 控制台创建集群，请进入到 **Infrastructure > Clusters**。在 **Clusters** 页面上，点 **Create cluster** 并完成控制台中的步骤。

注：此过程用于创建集群。如果您有一个要导入的现有集群，请参阅[将目标受管集群导入到 hub 集群](#)以了解这些步骤。

如果您需要创建凭证，请参阅[Red Hat OpenStack Platform 创建凭证](#)。

集群的主机名用于集群的主机名。名称必须包含少于 15 个字符。这个值必须与创建凭证先决条件中列出的 DNS 记录的名称匹配。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要将集群添加到现有的集群集中，则需要为集群设置具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与您为 **Red Hat OpenStack Platform** 帐户配置的所选凭证关联的基本 DNS 域，则该值会在字段中填充。您可以修改它的值来覆盖它。如需更多信息，请参阅 **Red Hat OpenStack Platform** 文档中的[管理域](#)。此名称用于集群的主机名。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 URL。[发行镜像](#) 以了解有关发行镜像的更多信息。仅支持 **OpenShift Container Platform** 版本 4.6.x 或更高版本的发行镜像。

节点池包括 **control plane** 池和 **worker** 池。**control plane** 节点共享集群活动的管理。如果受管集群的构架类型与 **hub** 集群的构架不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**, **ppc64le**, **s390x**, 和 **arm64**。

您必须为 **control plane** 池添加实例类型，但您可以在创建后更改实例的类型和大小。

您可以在 **worker** 池中创建一个或多个 **worker** 节点，以运行集群的容器工作负载。它们可以位于单个 **worker** 池中，也可以分布在多个 **worker** 池中。如果指定了零个 **worker** 节点，**control plane** 节点也充当 **worker** 节点。该信息包括以下字段：

- **实例类型**：您可以在实例创建后更改实例的类型和大小。
- **节点数**：指定 **worker** 池的节点数。定义 **worker** 池时需要此设置。

集群需要网络详情。您必须为 **IPv4** 网络提供一个或多个网络的值。对于 **IPv6** 网络，您必须定义多个网络。

您可以通过点 **Add network** 来添加额外网络。如果您使用的是 **IPv6** 地址，您必须有多个网络。

凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样，覆盖这些信息，或者在要启用代理时添加信息。以下列表包含创建代理所需的信息：

- **HTTP proxy**：指定应用作 **HTTP** 流量的代理的 **URL**。
- **HTTPS 代理**：用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则与 **HTTP** 代理相同的值用于 **HTTP** 和 **HTTPS**。
- **no proxy**：定义应绕过代理的站点的逗号分隔列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。
- **其他信任捆绑包**：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

您可以点 **Disconnected** 安装来定义断开连接的安装镜像。当使用 **Red Hat OpenStack Platform** 供应商和断开连接的安装创建集群时，如果需要一个证书才能访问镜像 **registry**，在配置集群时需要在 **Configuration for disconnected installation** 段中的 **Additional trust bundle** 字段中输入它，在创建集群时在 **Disconnected installation** 段中输入它。

当您在创建集群前查看信息并选择性地自定义它时，您可以点 **YAML** 切换 **On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。

在创建使用内部证书颁发机构(CA)的集群时，您需要通过完成以下步骤为集群自定义 **YAML** 文件：

1.

使用 **review** 步骤上的 **YAML** 开关，使用 **CA** 证书捆绑包在列表的顶部插入 **Secret** 对象。注：如果 **Red Hat OpenStack Platform** 环境使用由多个机构签名的证书提供服务，则捆绑包必须包含用于验证所有所需端点的证书。为名为 **ocp3** 的集群添加类似于以下示例：

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: ocp3-openstack-trust
  namespace: ocp3
stringData:
  ca.crt: |
    -----BEGIN CERTIFICATE-----
    <Base64 certificate contents here>
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    <Base64 certificate contents here>
    -----END CERTIFICATE-----
```

2.

修改 **Hive ClusterDeployment** 对象，在 **spec.platform.openstack** 中指定 **certificate SecretRef** 的值，如下例所示：

```
platform:
  openstack:
    certificatesSecretRef:
      name: ocp3-openstack-trust
    credentialsSecretRef:
      name: ocp3-openstack-creds
  cloud: openstack
```

前面的示例假定 **clouds.yaml** 文件中的云名称为 **openstack**。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine operator** 管理自动配置。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.9. 在 Red Hat Virtualization 上创建集群

您可以使用多集群引擎 **operator** 控制台在 **Red Hat Virtualization** 上创建 **Red Hat OpenShift**

Container Platform 集群。

在创建集群时，创建过程使用 **Hive** 资源中的 **OpenShift Container Platform** 安装程序。如果在完成此步骤后集群创建有疑问，请参阅 **OpenShift Container Platform** 文档中的在 [RHV 上安装](#) 有关流程的更多信息。

- [先决条件](#)
- [使用控制台创建集群](#)

1.5.4.9.1. 先决条件

在 **Red Hat Virtualization** 上创建集群前请查看以下先决条件：

- 您必须已部署 **hub** 集群。
- 您需要 **Red Hat Virtualization** 凭证。如需更多信息，请参阅为 **Red Hat Virtualization** 创建凭证。
- 您需要 **oVirt Engine** 虚拟机配置的域和虚拟机代理。有关如何配置域的说明，请参阅 **Red Hat OpenShift Container Platform** 文档中的在 [RHV 上安装](#)。
- 您必须具有 **Red Hat Virtualization** 登录凭证，其中包括您的红帽客户门户网站用户名和密码。
- 您需要 **OpenShift Container Platform** 镜像 **pull secret**。您可以从以下位置下载 **pull secret**：[Pull secret](#)。如需有关 **pull secret** 的更多信息，请参阅使用镜像 **pull secret**。

注：如果您在云供应商上更改云供应商访问密钥，则需要在 **multicluster engine operator** 的控制台中手动更新云供应商的对应凭证。当凭证在托管受管集群的云供应商过期并尝试删除受管集群时，需要此项。

1.5.4.9.2. 使用控制台创建集群

要从 **multicluster engine operator** 控制台创建集群，请进入到 **Infrastructure > Clusters**。在 **Clusters** 页面上，点 **Create cluster** 并完成控制台中的步骤。

注：此过程用于创建集群。如果您有一个要导入的现有集群，请参阅[将目标受管集群导入到 hub 集群](#)以了解这些步骤。

如果您需要创建凭证，请参阅[为 Red Hat Virtualization 创建凭证](#)。

集群名称用于集群的主机名。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要将集群添加到现有的集群集中，则需要在集群设置上具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与您为 **Red Hat Virtualization** 帐户配置的所选凭证关联的基本 **DNS** 域，则该值会在那个字段中填充。您可以覆盖该值来更改它。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 **URL**。[发行镜像](#) 以了解有关发行镜像的更多信息。

节点池的信息包括 **control plane** 池的内核、插槽、内存和磁盘大小。这三个 **control plane** 节点共享集群活动的管理。该信息包括 **Architecture** 字段。查看以下字段描述：

- **CPU 架构**：如果受管集群的架构类型与 **hub** 集群的架构不同，请为池中机器的说明集合架构输入一个值。有效值为 **amd64**, **ppc64le**, **s390x**, 和 **arm64**。

worker 池信息需要池名称、内核数量、内存分配、磁盘大小分配和 **worker** 池的节点数。**worker** 池中的 **worker** 节点可以在单个 **worker** 池中，也可以分布到多个 **worker** 池中。

您预先配置的 **oVirt** 环境中需要以下网络详情。

- **ovirt 网络名称**

- **vNIC 配置文件 ID**：指定虚拟网卡配置文件 ID。

- **API VIP**：指定用于内部 **API** 通信的 **IP** 地址。

注：这个值必须与您用来创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。如果没有提供，**DNS** 必须预先配置，以便 **api.** 可以正确解析。

- **Ingress VIP**：指定用于入口流量的 **IP** 地址。

注：这个值必须与您用来创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。如果没有提供，则必须预先配置 **DNS**，以便 **test.apps** 可以被正确解析。

- **Network type**：默认值为 **OpenShiftSDN**。**OVNKubernetes** 是使用 **IPv6** 的必要设置。

- **Cluster network CIDR**：此 **IP** 地址可用于 **pod IP** 地址的数量和列表。这个块不能与另一个网络块重叠。默认值为 **10.128.0.0/14**。

- **网络主机前缀**：为每个节点设置子网前缀长度。默认值为 **23**。

- **Service network CIDR**：为服务提供 **IP** 地址块。这个块不能与另一个网络块重叠。默认值为 **172.30.0.0/16**。

- **Machine CIDR**：提供 **OpenShift Container Platform** 主机使用的 **IP** 地址块。这个块不能与另一个网络块重叠。默认值为 **10.0.0.0/16**。

您可以通过点 **Add network** 来添加额外网络。如果您使用的是 **IPv6** 地址，您必须有多个网络。

凭证中提供的代理信息会自动添加到代理字段中。您可以使用信息原样，覆盖这些信息，或者在要启用代理时添加信息。以下列表包含创建代理所需的信息：

- **HTTP proxy**：指定应用作 **HTTP** 流量的代理的 **URL**。
- **HTTPS 代理**：指定用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- **没有代理站点**：提供以逗号分隔的站点列表，这些站点应绕过代理。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。
- **其他信任捆绑包**：代理 **HTTPS** 连接所需的一个或多个额外 **CA** 证书。

当您在创建集群前查看信息并选择性地自定义它时，您可以点 **YAML 切换 On** 查看面板中的 **install-config.yaml** 文件内容。如果有更新，您可以使用自定义设置编辑 **YAML** 文件。

注：您不必运行随集群提供的集群详情的 **oc** 命令。创建集群时，它由 **multicluster engine operator** 管理自动配置。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.10. 在内部环境中创建集群

您可以使用控制台创建内部 **Red Hat OpenShift Container Platform** 集群。

您可以使用此流程在 **VMware vSphere**、**Red Hat OpenStack**、**Red Hat Virtualization Platform** 和裸机环境中创建单节点 **OpenShift (SNO)** 集群、多节点集群和紧凑的三节点集群。

没有与安装集群的平台集成，因为平台值设置为 **platform=none**。单节点 **OpenShift** 集群仅包含一个节点，用于托管 **control plane** 服务和用户工作负载。当您最小化集群资源占用空间时，此配置很有

用。

您还可以使用零接触置备功能在边缘资源上置备多个单节点 **OpenShift** 集群，该功能可通过 **Red Hat OpenShift Container Platform** 使用。如需更多信息，请参阅 **OpenShift Container Platform** 文档中的 [在断开连接的环境中大规模部署分布式单元](#)。

- [先决条件](#)
- [使用控制台创建集群](#)
- [使用命令行创建集群](#)

1.5.4.10.1. 先决条件

在内部环境中创建集群前需要满足以下先决条件：

- 您必须在 **OpenShift Container Platform** 版本 **4.9** 或更高版本上部署了 **hub** 集群。
- 您需要配置了主机清单的基础架构环境。
- 您必须对 **hub** 集群（连接）或连接到互联网（断开连接）的内部或镜像 **registry** 的连接，以检索用于创建集群所需的镜像。
- 您需要配置了内部凭证。
- 您需要 **OpenShift Container Platform** 镜像 **pull secret**。如需更多信息，请参阅[使用镜像 pull secret](#)。

1.5.4.10.2. 使用控制台创建集群

要从控制台创建集群，请导航到 **Infrastructure > Clusters**。在 **Clusters** 页面上，点 **Create cluster** 并完成控制台中的步骤。

- 选择 **Host inventory** 作为集群的类型。

以下选项可用于您的支持安装：

- 使用现有发现的主机：从现有主机清单中的主机列表中选择您的主机。
- 发现新主机：发现不在现有基础架构环境中的主机。发现您自己的主机，而不是使用已在基础架构环境中的主机。

如果您需要创建凭证，请参阅 [为内部环境创建凭证](#)。

集群的名称用于集群的主机名。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

提示：在控制台中输入信息时，选择 **YAML: On** 查看内容更新。

如果要将集群添加到现有的集群集中，则需要在集群设置上具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群集选项，请联络您的集群管理员，为集群集提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

如果已有与您为供应商帐户配置的所选凭证关联的基本 **DNS** 域，则该值会在那个字段中填充。您可以通过覆盖它来更改值，但创建集群后无法更改此设置。此基础域用于创建到 **OpenShift Container Platform** 集群组件的路由。它在集群供应商的 **DNS** 中被配置为授权起始(SOA)记录。

OpenShift 版本 标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。如果要使用的版本可用，您可以从镜像列表中选择镜像。如果您要使用的镜像不是标准镜像，您可以输入您要使用的镜像的 **URL**。[发行镜像](#) 以了解有关发行镜像的更多信息。

当您选择 **4.9** 或更高版本的 **OpenShift** 版本时，会显示 **Install single node OpenShift(SNO)** 的选项。单节点 **OpenShift** 集群仅包含一个节点，用于托管 **control plane** 服务和用户工作负载。

如果您希望集群是一个单节点 **OpenShift** 集群，请选择 **单节点 OpenShift** 选项。您可以通过完成以下步骤，向单节点 **OpenShift** 集群添加额外的 **worker**：

1. 在控制台中，导航到 **Infrastructure > Clusters**，再选择您创建的或想要访问的集群名称。
2. 选择 **Actions > Add hosts** 来添加额外的 **worker**。

注：单节点 **OpenShift control plane** 需要 **8** 个 **CPU** 内核，而多节点 **control plane** 集群的 **control plane** 节点只需要 **4** 个 **CPU** 内核。

查看并保存集群后，您的集群将保存为集群草稿。您可以通过在 **Clusters** 页面中选择集群名称来关闭创建过程，并在稍后完成该过程。

如果您使用的是现有主机，请选择是否要自行选择主机，还是自动选择它们。主机数量取决于您选择的节点数量。例如，**SNO** 集群只需要一个主机，而标准的三节点集群需要三个主机。

主机位置列表中显示了满足此集群要求的可用主机位置。对于主机的分发和更加高可用性的配置，请选择多个位置。

如果您要发现没有现有基础架构环境的新主机，请使用 **Discovery Image** 完成将主机添加到主机清单中的步骤。

绑定主机以及验证通过后，通过添加以下 **IP** 地址完成集群的网络信息：

- **API VIP**：指定用于内部 **API** 通信的 **IP** 地址。

注：这个值必须与您用来创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。如果没有提供，**DNS** 必须预先配置，以便 **api.** 可以正确解析。

- **Ingress VIP** : 指定用于入口流量的 IP 地址。

注：这个值必须与您用来创建 **prerequisites** 部分中列出的 **DNS** 记录的名称匹配。如果没有提供，则必须预先配置 **DNS**，以便 **test.apps** 可以被正确解析。

您可以在 **Clusters** 导航页面中查看安装状态。

如需了解更多与访问集群相关的信息，继续[访问集群](#)。

1.5.4.10.3. 使用命令行创建集群

您还可以使用中央基础架构管理组件中的辅助安装程序功能在没有控制台的情况下创建集群。完成此步骤后，您可以从生成的发现镜像引导主机。步骤的顺序通常并不重要，但在有必要顺序时请注意。

1.5.4.10.3.1. 创建命名空间

需要一个资源的命名空间。将所有资源保留在共享命名空间中更为方便。本例使用 **sample-namespace** 作为命名空间的名称，但您可以使用除 **assisted-installer** 以外的任何名称。通过创建并应用以下文件来创建命名空间：

```
apiVersion: v1
kind: Namespace
metadata:
  name: sample-namespace
```

1.5.4.10.3.2. 将 pull secret 添加到命名空间

通过创建并应用以下自定义资源，将 **pull secret** 添加到您的命名空间中：

```
apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: <pull-secret>
  namespace: sample-namespace
stringData:
  .dockerconfigjson: 'your-pull-secret-json' ❶
```

❶

添加 **pull secret** 的内容。例如，这可以包括 **cloud.openshift.com**、**quay.io** 或 **registry.redhat.io** 身份验证。

1.5.4.10.3.3. 创建 ClusterImageSet

通过创建并应用以下自定义资源，生成 **CustomImageSet** 来为集群指定 **OpenShift Container Platform** 版本：

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: openshift-v4.12.0
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.12.0-rc.0-x86_64
```

1.5.4.10.3.4. 创建 ClusterDeployment 自定义资源

ClusterDeployment 自定义资源定义是一个控制集群生命周期的 **API**。它引用了定义集群参数的 **spec.ClusterInstallRef** 设置中的 **AgentClusterInstall** 自定义资源。

根据以下示例创建并应用 **ClusterDeployment** 自定义资源：

```
apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: single-node
  namespace: demo-worker4
spec:
  baseDomain: hive.example.com
  clusterInstallRef:
    group: extensions.hive.openshift.io
    kind: AgentClusterInstall
    name: test-agent-cluster-install ①
    version: v1beta1
  clusterName: test-cluster
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    agentBareMetal:
      agentSelector:
        matchLabels:
          location: internal
  pullSecretRef:
    name: <pull-secret> ②
```

①

2

使用您下载的 **pull secret** 将 **pull secret** 添加到命名空间。

1.5.4.10.3.5. 创建 AgentClusterInstall 自定义资源

在 **AgentClusterInstall** 自定义资源中，您可以指定集群的许多要求。例如，您可以指定集群网络设置、平台、**control plane** 数量和 **worker** 节点。

创建并添加类似以下示例的自定义资源：

```
apiVersion: extensions.hive.openshift.io/v1beta1
kind: AgentClusterInstall
metadata:
  name: test-agent-cluster-install
  namespace: demo-worker4
spec:
  platformType: BareMetal 1
  clusterDeploymentRef:
    name: single-node 2
  imageSetRef:
    name: openshift-v4.12.0 3
  networking:
    clusterNetwork:
      - cidr: 10.128.0.0/14
        hostPrefix: 23
    machineNetwork:
      - cidr: 192.168.111.0/24
    serviceNetwork:
      - 172.30.0.0/16
  provisionRequirements:
    controlPlaneAgents: 1
  sshPublicKey: ssh-rsa <your-public-key-here> 4
```

1

指定创建集群的环境的平台类型。有效值为 **BareMetal**、无、**VSphere** 或 **External**。

2

使用与 **ClusterDeployment** 资源相同的名称。

3

4

您可以指定 **SSH** 公钥，该密钥可让您在安装后访问主机。

1.5.4.10.3.6. 可选：创建 **NMStateConfig** 自定义资源

只有在有主机级别网络配置（如静态 **IP** 地址）时才需要 **NMStateConfig** 自定义资源。如果包含此自定义资源，则必须在创建 **InfraEnv** 自定义资源前完成此步骤。**NMStateConfig** 由 **InfraEnv** 自定义资源中的 **spec.nmStateConfigLabelSelector** 的值引用。

创建并应用 **NMStateConfig** 自定义资源，如下例所示。根据需要替换值：

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: <mynmstateconfig>
  namespace: <demo-worker4>
  labels:
    demo-nmstate-label: <value>
spec:
  config:
    interfaces:
      - name: eth0
        type: ethernet
        state: up
        mac-address: 02:00:00:80:12:14
        ipv4:
          enabled: true
          address:
            - ip: 192.168.111.30
              prefix-length: 24
          dhcp: false
      - name: eth1
        type: ethernet
        state: up
        mac-address: 02:00:00:80:12:15
        ipv4:
          enabled: true
          address:
            - ip: 192.168.140.30
              prefix-length: 24
          dhcp: false
    dns-resolver:
      config:
        server:
          - 192.168.126.1
    routes:
      config:
```

```

- destination: 0.0.0/0
  next-hop-address: 192.168.111.1
  next-hop-interface: eth1
  table-id: 254
- destination: 0.0.0/0
  next-hop-address: 192.168.140.1
  next-hop-interface: eth1
  table-id: 254
interfaces:
- name: "eth0"
  macAddress: "02:00:00:80:12:14"
- name: "eth1"
  macAddress: "02:00:00:80:12:15"

```

注：您必须在 **InfraEnv** 资源 `spec.nmStateConfigLabelSelector.matchLabels` 字段中包含 `demo-nmstate-label` 标签名称和值。

1.5.4.10.3.7. 创建 **InfraEnv** 自定义资源

InfraEnv 自定义资源提供创建发现 **ISO** 的配置。在这个自定义资源中，您可以识别代理设置、**ignition overrides** 并指定 **NMState** 标签的值。此自定义资源中的 `spec.nmStateConfigLabelSelector` 值引用 **NMStateConfig** 自定义资源。

注：如果您计划包含可选的 **NMStateConfig** 自定义资源，您必须在 **InfraEnv** 自定义资源中引用它。如果您在创建 **NMStateConfig** 自定义资源前创建 **InfraEnv** 自定义资源，请编辑 **InfraEnv** 自定义资源来引用 **NMStateConfig** 自定义资源，并在添加引用后下载 **ISO**。

创建并应用以下自定义资源：

```

apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: myinfraenv
  namespace: demo-worker4
spec:
  clusterRef:
    name: single-node ①
    namespace: demo-worker4 ②
  pullSecretRef:
    name: pull-secret
    sshAuthorizedKey: <your_public_key_here> ③
  nmStateConfigLabelSelector:
    matchLabels:
      demo-nmstate-label: value ④

```

①

2

替换 [Create the ClusterDeployment](#) 中的 `clusterDeployment` 资源命名空间。

3

4

标签名称和标签值必须与您在 [Optional: Create the NMStateConfig](#) 自定义资源中创建的 `NMStateConfig` 自定义资源的 `label` 部分中的值匹配。

1.5.4.10.3.8. 从发现镜像引导主机

剩余的步骤解释了如何从之前的步骤发现 **ISO** 镜像引导主机。

1. 运行以下命令，从命名空间下载发现镜像：

```
curl --insecure -o image.iso $(kubectl -n sample-namespace get infraenvs.agent-install.openshift.io myinfraenv -o=jsonpath="{.status.isoDownloadURL}")
```

2. 将发现镜像移到虚拟介质、**USB** 驱动器或其他存储位置，并从您下载的发现镜像引导主机。

3. **Agent** 资源会自动创建。它注册到集群，并代表从发现镜像引导的主机。运行以下命令来批准 **Agent** 自定义资源并启动安装：

```
oc -n sample-namespace patch agents.agent-install.openshift.io 07e80ea9-200c-4f82-aff4-4932acb773d4 -p '{"spec":{"approved":true}}' --type merge
```

将代理名称和 **UUID** 替换为您的值。

当上一命令的输出包含目标集群的条目时，您可以确认它已被批准，其中包括 **APPROVED** 参数的值为 **true**。

1.5.4.11. 休眠创建的集群（技术预览）

您可以休眠使用 **multicluster engine operator** 创建的集群来节省资源。休眠集群需要的资源比正在运行的资源要少得多，因此您可以通过将集群移入和停止休眠状态来降低供应商成本。此功能只适用于在以下环境中由 **multicluster engine operator** 创建的集群：

- **Amazon Web Services**
- **Microsoft Azure**
- **Google Cloud Platform**

1.5.4.11.1. 使用控制台休眠集群

要使用控制台休眠由多集群引擎 **operator** 创建的集群，请完成以下步骤：

1. 在导航菜单中选择 **Infrastructure > Clusters**。确保已选中 **Manage cluster** 选项卡。
2. 从集群的 **Options** 菜单中选择 **Hibernate cluster**。注：如果 **Hibernate cluster** 选项不可用，您就无法休眠该群集。当集群导入且不是由 **multicluster engine operator** 创建时，会出现这种情况。

当进程完成后，**Clusters** 页面中的集群状态为 **Hibernating**。

提示：您可以通过在 **Clusters** 页面上选择要休眠的集群并选择 **Actions > Hibernate clusters** 来休眠多个集群。

您选择的集群正在休眠。

1.5.4.11.2. 使用 CLI Hibernate 集群

要使用 **CLI** 休眠由多集群引擎 **operator** 创建的集群，请完成以下步骤：

1. 输入以下命令编辑您要休眠的集群设置：

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

将 **name-of-cluster** 替换为您要休眠的集群的名称。

将 **namespace-of-cluster** 替换为您要休眠的集群的命名空间。

2.

将 **spec.powerState** 的值改为 **Hibernating**。

3.

输入以下命令查看集群的状态：

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

将 **name-of-cluster** 替换为您要休眠的集群的名称。

将 **namespace-of-cluster** 替换为您要休眠的集群的命名空间。

当集群休眠过程完成时，集群的 **type** 值为 **type=Hibernating**。

您选择的集群正在休眠。

1.5.4.11.3. 使用控制台恢复休眠集群的一般操作

要使用控制台恢复休眠集群的一般操作，请完成以下步骤：

1.

在导航菜单中选择 **Infrastructure > Clusters**。确保已选中 **Manage cluster** 选项卡。

2.

从您要恢复的集群的 **Options** 菜单中选择 **Resume cluster**。

当进程完成后，**Clusters** 页面中的集群状态为 **Ready**。

提示：您可以通过在 **Clusters** 页面上选择要恢复的集群并选择 **Actions > Resume clusters** 来休眠

多个集群。

所选集群恢复正常操作。

1.5.4.11.4. 使用 CLI 恢复休眠集群的一般操作

要使用 CLI 恢复休眠集群的一般操作，请完成以下步骤：

1.

输入以下命令来编辑集群的设置：

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

将 **name-of-cluster** 替换为您要休眠的集群的名称。

将 **namespace-of-cluster** 替换为您要休眠的集群的命名空间。

2.

将 **spec.powerState** 的值改为 **Running**。

3.

输入以下命令查看集群的状态：

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

将 **name-of-cluster** 替换为您要休眠的集群的名称。

将 **namespace-of-cluster** 替换为您要休眠的集群的命名空间。

完成恢复集群的过程后，集群的 **type** 值为 **type=Running**。

所选集群恢复正常操作。

1.5.4.12. 在代理环境中创建集群

当 **hub** 集群通过代理服务器连接时，您可以创建 **Red Hat OpenShift Container Platform** 集群。要成功创建集群，则必须满足以下情况之一：

- **multicluster engine operator** 具有与您要创建的受管集群连接的私有网络连接，受管集群使用代理访问互联网。
- 受管集群位于基础架构供应商上，但防火墙端口启用了从受管集群到 **hub** 集群的通信。

要创建使用代理配置的集群，请完成以下步骤：

1.

通过将以下信息添加到存储于您的 **Secret** 中的 **install-config YAML** 中添加以下信息，在 **hub** 集群上配置 **cluster-wide-proxy** 设置：

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

使用代理服务器的用户名替换 **username**。

使用密码替换 **password** 以访问您的代理服务器。

将 **proxy.example.com** 替换为代理服务器的路径。

使用与代理服务器的通信端口替换 **port**。

将 **wildcard-of-domain** 替换为应当绕过代理的域的条目。

使用置备网络的 **IP** 地址和分配的 **IP** 地址（以 **CIDR** 表示）替换 **provisioning-network/CIDR**。

将 **BMC-address-range/CIDR** 替换为 **BMC** 地址和地址数（以 **CIDR** 表示）。

添加前面的值后，设置将应用到集群。

2.

通过完成创建集群的步骤来置备集群。请参阅[创建集群](#)以选择您的供应商。

注：在部署集群时，只能使用 **install-config YAML**。部署集群后，您对 **install-config YAML** 所做的任何新更改都不适用。要在部署后更新配置，您必须策略。如需更多信息，请参阅 [Pod 策略](#)。

1.5.5. 将目标受管集群导入到 **hub** 集群

您可以从不同的 **Kubernetes** 云供应商导入集群。导入后，目标集群将变为 **multicluster engine operator hub** 集群的受管集群。除非另有指定，否则请在可以访问 **hub** 集群和目标受管集群的任何位置完成导入任务。

hub 集群无法管理任何其他 **hub** 集群，但可以管理自己。**hub** 集群被配置为自动导入和自助管理。您不需要手动导入 **hub** 集群。

但是，如果您删除 **hub** 集群并尝试再次导入它，则需要将 **local-cluster:true** 标签添加到 **ManagedCluster** 资源中。

从以下说明中选择来设置受管集群：

所需的用户类型或访问权限级别：集群管理员

- [使用控制台导入现有集群](#)
- [使用 CLI 导入受管集群](#)
- [导入内部 Red Hat OpenShift Container Platform 集群](#)

1.5.5.1. 使用控制台导入受管集群

为 **Kubernetes operator** 安装多集群引擎后，就可以导入集群来管理。继续阅读以下主题以了解如何使用控制台导入受管集群：

- [先决条件](#)
- [创建新的 **pull secret**](#)
- [导入集群](#)
- [手动运行导入命令的额外步骤](#)
- [可选：配置集群 **API** 地址](#)
- [删除集群](#)

1.5.5.1.1. 先决条件

- 已部署 **hub** 集群。如果要导入裸机集群，必须在 **Red Hat OpenShift Container Platform** 版本 **4.8** 或更高版本上安装 **hub** 集群。
- 要管理的集群。
- **base64** 命令行工具。
- 如果您导入不是由 **OpenShift Container Platform** 创建的集群，则定义的 **multiclusterhub.spec.imagePullSecret**。安装 **Kubernetes operator** 的多集群引擎时，可能会创建此 **secret**。如需有关如何定义此 **secret** 的更多信息，请参阅自定义镜像 **pull secret**。

所需的用户类型或访问权限级别：集群管理员

1.5.5.1.2. 创建新的 **pull secret**

如果需要创建新的 **pull secret**，请完成以下步骤：

1. 从 cloud.redhat.com 下载 **Kubernetes pull secret**。
2. 将 **pull secret** 添加到 **hub** 集群的命名空间。
3. 运行以下命令在 **open-cluster-management** 命名空间中创建新 **secret**：

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

将 **open-cluster-management** 替换为 **hub** 集群的命名空间的名称。**hub** 集群的默认命名空间是 **open-cluster-management**。

将 **path-to-pull-secret** 替换为您下载的 **pull secret** 的路径。

在导入时，**secret** 会自动复制到受管集群。

- 确保从您要导入的集群中删除之前安装的代理。您必须删除 **open-cluster-management-agent** 和 **open-cluster-management-agent-addon** 命名空间以避免错误。
- 有关在 **Red Hat OpenShift Dedicated** 环境中导入，请参阅以下备注：
 - 您必须在 **Red Hat OpenShift Dedicated** 环境中部署了 **hub** 集群。
 - **Red Hat OpenShift Dedicated** 的默认权限是 **dedicated-admin**，但不包含创建命名空间的所有权限。您必须具有 **cluster-admin** 权限才能导入和管理使用 **multicluster engine operator** 的集群。

1.5.5.1.3. 导入集群

您可以从控制台为每个可用的云供应商导入现有集群。

注：hub 集群无法管理不同的 hub 集群。hub 集群被设置为自动导入和管理自身，因此您不必手动导入 hub 集群来管理自己。

默认情况下，命名空间用于集群名称和命名空间，但您可以更改它。

重要：当您创建集群时，控制器会为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，集群会自动添加到默认受管集群集中。

如果要将集群添加到不同的集群集中，则必须对集群集具有 **clusterset-admin** 权限。如果在导入集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定集群集中没有正确的权限，集群导入会失败。如果没有要选择的集群设置选项，请联系集群管理员，为集群集提供 **clusterset-admin** 权限。

如果您导入了 **OpenShift Container Platform Dedicated** 集群，且没有为 **vendor=OpenShiftDedicated** 指定一个标签，或者为 **vendor=auto-detect** 添加标签，则 **managed-by=platform** 标签会自动添加到集群中。您可以使用此添加的标签将集群识别为 **OpenShift Container Platform Dedicated** 集群，并作为一个组检索 **OpenShift Container Platform Dedicated** 集群。

下表提供了 导入模式 的可用选项，它指定了导入集群的方法：

手动运行导入命令	在控制台中完成并提交信息后，包括任何 Red Hat Ansible Automation Platform 模板，在目标集群上运行提供的命令以导入集群。
为现有集群输入服务器 URL 和 API 令牌	提供您要导入的集群的服务器 URL 和 API 令牌。您可以指定一个 Red Hat Ansible Automation Platform 模板，以便在集群升级时运行。
提供 kubeconfig 文件	复制并粘贴您要导入的集群的 kubeconfig 文件的内容。您可以指定一个 Red Hat Ansible Automation Platform 模板，以便在集群升级时运行。

注：您必须已从 **OperatorHub** 安装 **Red Hat Ansible Automation Platform Resource Operator**，以创建并运行 **Ansible Automation Platform** 作业。要配置集群 API 地址，请参阅 [可选：配](#)

置集群 API 地址

1.5.5.1.3.1. 手动运行导入命令的额外步骤

这个方法要求您在目标集群上运行提供的命令来导入集群。

重要：命令中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。考虑在 <https://cloud.redhat.com/> 创建一个二级 **pull secret**，或创建一个服务帐户来保护个人凭证。

1. 生成并复制导入命令：
 - a. 从控制台导航中选择 **Infrastructure > Clusters**。
 - b. 选择 **Import cluster**。
 - c. 以表单填写 **Details** 和 **Automation** 信息，将 **Import mode** 选择设置为 **手动运行导入命令**。点 **Generate** 命令后，会显示集群概览页面。
 - d. 点 **Copy** 命令将生成的命令复制到剪贴板。
2. 对于 **OpenShift Container Platform Dedicated** 环境，请完成以下步骤（如果您不使用 **OpenShift Container Platform Dedicated** 环境，请跳至第 3 步）：
 - a. 登录到您要导入的集群的 **OpenShift Container Platform** 控制台。
 - b. 在您要导入的集群中创建 **open-cluster-management-agent** 和 **open-cluster-management** 命名空间或项目。
 - c. 在 **OpenShift Container Platform** 目录中找到 **klusterlet Operator**。
 - d. 在 **open-cluster-management** 命名空间中或您创建的项目中安装它。

重要：不要在 **open-cluster-management-agent** 命名空间中安装 **Operator**。

e.

通过完成以下步骤，从导入命令中提取 **bootstrap secret**：

i.

将导入命令粘贴到您创建的名称为 **import-command** 的文件中。

ii.

运行以下命令以将内容插入新文件中：

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

iii.

在输出中找到并复制名称为 **bootstrap-hub-kubeconfig** 的 **secret**。

iv.

将 **secret** 应用到受管集群上的 **open-cluster-management-agent** 命名空间。

v.

使用安装的 **Operator** 中的示例创建 **klusterlet** 资源。将 **clusterName** 值更改为与导入过程中设置的集群名称相同的名称。

注：当 **managedcluster** 资源成功注册到 **hub** 时，会安装两个 **klusterlet operator**。一个 **klusterlet operator** 位于 **open-cluster-management** 命名空间中，另一个位于 **open-cluster-management-agent** 命名空间中。具有多个 **Operator** 不会影响 **klusterlet** 的功能。

3.

输入以下命令确保您的 **kubectl** 登录到您要导入的集群：

```
kubectl cluster-info
```

4.

运行您复制的命令，将 **open-cluster-management-agent-addon** 部署到受管集群。

如果要配置集群 **API** 地址，请按照 [可选的步骤进行：配置集群 API 地址](#)。

1.5.5.1.3.2. 可选：配置集群 API 地址

完成以下步骤，通过配置运行 **oc get managedcluster** 命令时表中显示的 **URL** 来选择性地配置集群

详情页面上的 **Cluster API** 地址：

1. 使用具有 **cluster-admin** 权限的 **ID** 登录到 **hub** 集群。
2. 为目标受管集群配置 **kubeconfig** 文件。
3. 运行以下命令，编辑您要导入的集群的受管集群条目：

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

将 **cluster-name** 替换为受管集群的名称。

4. 在 **YAML** 文件中的 **ManagedCluster spec** 中添加 **ManagedClusterClientConfigs**，如下例所示：

```
spec:
  hubAcceptsClient: true
  managedClusterClientConfigs:
    - url: <https://api.new-managed.dev.redhat.com>
```

将 **URL** 值替换为提供对您要导入的受管集群的外部访问的 **URL**。

1.5.5.1.4. 删除导入的集群

完成以下步骤以删除导入的集群以及在受管集群上创建的 **open-cluster-management-agent-addon**。

在 **Clusters** 页面上，点 **Actions > Detach cluster** 从管理中删除集群。

注意：如果您试图分离名为 **local-cluster** 的 **hub** 集群，请注意 **disableHub selfManagement** 的默认设置为 **false**。此设置会导致 **hub** 集群在分离时会重新导入自己并管理自己，并协调 **MultiClusterHub** 控制器。**hub** 集群可能需要几小时时间来完成分离过程并重新导入。如果要在等待进程完成的情况下重新导入 **hub** 集群，您可以运行以下命令来重启 **multiclusterhub-operator pod** 并更快地重新导入：

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

您可以通过将 `disableHubSelfManagement` 值改为 `true` 来更改 `hub` 集群的值，使其不会自动导入。如需更多信息，请参阅 [disableHubSelfManagement](#) 主题。

1.5.5.1.4.1. 其他资源

- 如需有关如何定义 [自定义镜像 pull secret](#) 的更多信息，请参阅 [自定义镜像 pull secret](#)。
- 请参阅 [disableHubSelfManagement](#) 主题。

1.5.5.2. 使用 CLI 导入受管集群

为 **Kubernetes operator** 安装多集群引擎后，就可以导入集群并使用 **Red Hat OpenShift Container Platform CLI** 管理它。继续阅读以下主题，了解如何使用自动导入 `secret` 或使用手动命令通过 **CLI** 导入受管集群。

- [先决条件](#)
- [支持的构架](#)
- [准备集群导入](#)
- [使用自动导入 `secret` 功能导入集群](#)
- [手动导入集群](#)
- [导入 `klusterlet` 附加组件](#)
- [使用 **CLI** 删除导入的集群](#)

重要： `hub` 集群无法管理不同的 `hub` 集群。`hub` 集群被设置为自动导入和管理自己作为 `本地集群`。您不必手动导入 `hub` 集群来自管理。如果您删除了 `hub` 集群并尝试再次导入它，则需要添加 `local-cluster:true` 标签。

1.5.5.2.1. 先决条件

- 已部署 **hub** 集群。如果要导入裸机集群，必须在 **OpenShift Container Platform** 版本 4.6 或更高版本上安装 **hub** 集群。
- 要管理的单独集群。
- **OpenShift Container Platform CLI** 版本 4.6 或更高版本来运行 **oc** 命令。如需有关安装和配置 **OpenShift Container Platform CLI** 的信息，请参阅 [OpenShift CLI 入门](#)。
- 如果您导入不是由 **OpenShift Container Platform** 创建的集群，则定义的 **multiclusterhub.spec.imagePullSecret**。安装 **Kubernetes operator** 的多集群引擎时，可能会创建此 **secret**。如需有关如何定义此 **secret** 的更多信息，请参阅[自定义镜像 pull secret](#)。

1.5.5.2.2. 支持的构架

- **Linux (x86_64, s390x, ppc64le)**
- **macOS**

1.5.5.2.3. 准备集群导入

在使用 **CLI** 导入受管集群前，您必须完成以下步骤：

1. 运行以下命令登录到您的 **hub** 集群：

```
oc login
```

2. 在 **hub** 集群上运行以下命令以创建项目和命名空间。在 `< cluster_name >` 中定义的集群名称也用作 **YAML** 文件和命令中的集群命名空间：

```
oc new-project <cluster_name>
```

重要： `cluster.open-cluster-management.io/managedCluster` 标签会自动添加到受管集群命名空间中并从中删除。不要手动将其添加到受管集群命名空间中或从受管集群命名空间中删除。

3.

使用以下示例内容，创建一个名为 **managed-cluster.yaml** 的文件：

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

当 **cloud** 和 **vendor** 的值被设置为 **auto-detect** 时，**Red Hat Advanced Cluster Management** 会检测您要导入的集群的云和厂商类型。您可以选择将 **auto-detect** 的值替换为集群的 **cloud** 和 **vendor** 值。请参见以下示例：

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

4.

运行以下命令，将 **YAML** 文件应用到 **ManagedCluster** 资源：

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

现在，您可以使用 [自动导入 secret](#) 或 [手动导入集群](#) 来继续导入集群。

1.5.5.2.4. 使用自动导入 **secret** 功能导入集群

要使用自动导入 **secret** 导入受管集群，您必须创建一个 **secret**，其中包含集群的 **kubeconfig** 文件，或 **kube API** 服务器和集群的令牌对。完成以下步骤，使用自动导入 **secret** 导入集群：

1.

检索您要导入的受管集群的 **kubeconfig** 文件或 **kube API** 服务器和令牌。请参阅 [Kubernetes 集群的文档](#)，了解在哪里可以找到您的 **kubeconfig** 文件或 **kube API** 服务器和令牌。

2.

在 **#{CLUSTER_NAME}** 命名空间中创建 **auto-import-secret.yaml** 文件。

a.

使用类似以下模板的内容，创建一个名为 **auto-import-secret.yaml** 的 **YAML** 文件：

```
apiVersion: v1
```

```

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

```

- b. 运行以下命令，在 `<cluster_name>` 命名空间中应用 **YAML** 文件：

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

注：默认情况下，自动导入 **secret** 会被使用一次，在导入过程完成后会被删除。如果要保留自动导入 **secret**，请将 **managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret** 添加到 **secret**。您可以运行以下命令来添加它：

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

3. 验证您的导入集群的 **JOINED** 和 **AVAILABLE** 状态。在 **hub** 集群中运行以下命令：

```
oc get managedcluster <cluster_name>
```

4. 在集群中运行以下命令来登录到受管集群：

```
oc login
```

5. 您可以运行以下命令来验证您要导入的集群中的 **pod** 状态：

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

现在，您可以继续导入 **klusterlet** 附加组件。

1.5.5.2.5. 手动导入集群

重要： 导入命令包含复制到每个导入的受管集群中的 **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。

完成以下步骤以手动导入受管集群：

1. 运行以下命令，获取由导入控制器在 **hub** 集群上生成的 **klusterlet-crd.yaml** 文件：

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

2. 运行以下命令，获取导入控制器在 **hub** 集群上生成的 **import.yaml** 文件：

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

在要导入的集群中执行以下步骤：

3. 输入以下命令登录到您导入的受管集群：

```
oc login
```

4. 运行以下命令应用您在第 **1** 步中生成的 **klusterlet-crd.yaml**：

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

5. 运行以下命令应用您之前生成的 **import.yaml** 文件：

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

6. 您可以从 **hub** 集群中运行以下命令来验证您要导入的受管集群的 **JOINED** 和 **AVAILABLE** 状态：

```
oc get managedcluster <cluster_name>
```

现在，您可以继续导入 **klusterlet** 附加组件。

1.5.5.2.6. 导入 klusterlet 附加组件

实现 **KlusterletAddonConfig klusterlet** 附加组件配置，以在受管集群上启用其他附加组件。通过完成以下步骤来创建并应用配置文件：

1. 创建一个类似以下示例的 **YAML** 文件：

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

2. 将文件保存为 **klusterlet-addon-config.yaml**。
3. 运行以下命令来应用 **YAML**：

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

附加组件会在您导入的受管集群状态变为 **AVAILABLE** 后安装。

4. 您可以运行以下命令来验证您要导入的集群中的附加组件的 **pod** 状态：

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

1.5.5.2.7. 使用命令行界面删除导入的集群

要使用命令行界面删除受管集群，请运行以下命令：

```
oc delete managedcluster <cluster_name>
```

将 **<cluster_name >** 替换为集群的名称。

1.5.5.3. 手动导入内部 Red Hat OpenShift Container Platform 集群

为 **Kubernetes operator** 安装多集群引擎后，就可以导入集群来管理。您可以导入现有的 **OpenShift Container Platform** 集群，以便您可以添加额外的节点。继续阅读以下主题以了解更多信息：

- [先决条件](#)
- [导入集群](#)
- [在 OpenShift Container Platform 集群中添加 worker 节点](#)

1.5.5.3.1. 先决条件

- [启用中央基础架构管理服务。](#)

1.5.5.3.2. 导入集群

完成以下步骤，手动导入 **OpenShift Container Platform** 集群，没有静态网络或裸机主机，并准备好添加节点：

1. 通过应用以下 **YAML** 内容，为要导入的 **OpenShift Container Platform** 集群创建一个命名空间：

```
apiVersion: v1
kind: Namespace
metadata:
  name: managed-cluster
```

2. 通过应用以下 **YAML** 内容，确保与您导入的 **OpenShift Container Platform** 集群匹配的 **ClusterImageSet** 已存在：

```

apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: openshift-v4.11.18
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-
  release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce
  7863

```

3.

通过应用以下 **YAML** 内容来添加 **pull secret** 以访问镜像：

```

apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: pull-secret
  namespace: managed-cluster
stringData:
  .dockerconfigjson: <pull-secret-json> 1

```

1

将 **<pull-secret-json>** 替换为您的 **pull secret JSON**。

4.

将 **kubeconfig** 从 **OpenShift Container Platform** 集群复制到 **hub** 集群。

a.

运行以下命令，从 **OpenShift Container Platform** 集群获取 **kubeconfig**。确保将 **kubeconfig** 设置为要导入的集群：

```

oc get secret -n openshift-kube-apiserver node-kubeconfigs -ojson | jq '.data["lb-
ext.kubeconfig"]' --raw-output | base64 -d > /tmp/kubeconfig.some-other-cluster

```

b.

运行以下命令，将 **kubeconfig** 复制到 **hub** 集群。确保将 **kubeconfig** 设置为 **hub** 集群：

```

oc -n managed-cluster create secret generic some-other-cluster-admin-kubeconfig --
from-file=kubeconfig=/tmp/kubeconfig.some-other-cluster

```

5.

通过应用以下 **YAML** 内容来创建 **AgentClusterInstall** 自定义资源。根据需要替换值：

```

apiVersion: extensions.hive.openshift.io/v1beta1

```

```

kind: AgentClusterInstall
metadata:
  name: <your-cluster-name> 1
  namespace: <managed-cluster>
spec:
  networking:
    userManagedNetworking: true
  clusterDeploymentRef:
    name: <your-cluster>
  imageSetRef:
    name: openshift-v4.11.18
  provisionRequirements:
    controlPlaneAgents: 2
  sshPublicKey: <""> 3

```

1

为集群选择一个名称。

2

如果使用单节点 **OpenShift** 集群，请使用 1。如果您使用多节点集群，请使用 3。

3

添加可选的 **sshPublicKey** 字段以登录到节点以进行故障排除。

6.

通过应用以下 **YAML** 内容来创建 **ClusterDeployment**：根据需要替换值：

```

apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: <your-cluster-name> 1
  namespace: managed-cluster
spec:
  baseDomain: <redhat.com> 2
  installed: <true> 3
  clusterMetadata:
    adminKubeconfigSecretRef:
      name: <your-cluster-name-admin-kubeconfig> 4
    clusterID: <""> 5
    infraID: <""> 6
  clusterInstallRef:
    group: extensions.hive.openshift.io
    kind: AgentClusterInstall
    name: your-cluster-name-install
    version: v1beta1
  clusterName: your-cluster-name
  platform:

```

```
agentBareMetal:
pullSecretRef:
  name: pull-secret
```

1

为集群选择一个名称。

2

确保 **baseDomain** 与用于 **OpenShift Container Platform** 集群的域匹配。

3

设置为 **true**，以自动将 **OpenShift Container Platform** 集群导入为生产环境集群。

4

引用在第 4 步中创建的 **kubeconfig**。

5 6

在生产环境中将 **clusterID** 和 **infraID** 留空。

7.

通过应用以下 **YAML** 内容，添加一个 **InfraEnv** 自定义资源来发现要添加到集群中的新主机。根据需要替换值：

注：如果您使用静态 **IP** 地址，以下示例可能需要其他配置。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: your-infraenv
  namespace: managed-cluster
spec:
  clusterRef:
    name: your-cluster-name
    namespace: managed-cluster
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: ""
```

表 1.5. InfraEnv 字段表

字段	可选或必需的	描述
clusterRef	选填	如果您使用 late 绑定，则 clusterRef 字段是可选的。如果没有使用更新的绑定，您必须添加 clusterRef 。
sshAuthorizedKey	选填	添加可选的 sshAuthorizedKey 字段以登录到节点以进行故障排除。

1. 如果导入成功，则会出现下载 ISO 文件的 URL。运行以下命令下载 ISO 文件，将 `<url>` 替换为显示的 URL：

注：您可以使用裸机主机自动执行主机发现。

```
oc get infraenv -n managed-cluster some-other-infraenv -ojson | jq ".status.<url>" --raw-output | xargs curl -k -o /storage0/isos/some-other.iso
```

1.5.5.3.3. 在 OpenShift Container Platform 集群中添加 worker 节点

完成以下步骤，将生产环境 worker 节点添加到 OpenShift Container Platform 集群中：

1. 从您之前下载的 ISO 中引导您要用作 worker 节点的机器。

注：确保 worker 节点满足 OpenShift Container Platform worker 节点的要求。

2. 运行以下命令后等待代理注册：

```
watch -n 5 "oc get agent -n managed-cluster"
```

3. 如果代理注册是 **successful**，则会列出代理。批准安装的代理。这可能需要几分钟时间。

注：如果没有列出代理，请按 **Ctrl** 和 **C** 退出 **watch** 命令，然后登录到 worker 节点以排除故障。

4. 如果使用更新的绑定，请运行以下命令将待处理的未绑定代理与 OpenShift Container Platform 集群关联。如果没有使用更新的绑定，请跳至第 5 步：

```
oc get agent -n managed-cluster -ojson | jq -r '.items[] | select(.spec.approved==false) | select(.spec.clusterDeploymentName==null) | .metadata.name' | xargs oc -n managed-cluster patch -p '{"spec":{"clusterDeploymentName":{"name":"some-other-cluster"},"namespace":"managed-cluster"}}' --type merge agent
```

5.

运行以下命令来批准任何待处理的代理以进行安装：

```
oc get agent -n managed-cluster -ojson | jq -r '.items[] | select(.spec.approved==false) | .metadata.name' | xargs oc -n managed-cluster patch -p '{"spec":{"approved":true}}' --type merge agent
```

等待 **worker** 节点的安装。当 **worker** 节点安装完成后，**worker** 节点使用证书签名请求(CSR)联系受管集群以开始加入过程。**CSR** 会自动签名。

1.5.5.4. 在受管集群中指定镜像 registry 进行导入

您可能需要覆盖您要导入的受管集群中的镜像 **registry**。您可以通过创建一个 **ManagedClusterImageRegistry** 自定义资源定义来实现。

ManagedClusterImageRegistry 自定义资源定义是一个命名空间范围的资源。

ManagedClusterImageRegistry 自定义资源定义为要选择的放置指定一组受管集群，但需要与自定义镜像 **registry** 不同的镜像。使用新镜像更新受管集群后，会在每个受管集群中添加以下标签进行识别：**open-cluster-management.io/image-registry=<namespace>**。**<managedClusterImageRegistryName>**。

以下示例显示了 **ManagedClusterImageRegistry** 自定义资源定义：

```
apiVersion: imageregistry.open-cluster-management.io/v1alpha1
kind: ManagedClusterImageRegistry
metadata:
  name: <imageRegistryName>
  namespace: <namespace>
spec:
  placementRef:
    group: cluster.open-cluster-management.io
    resource: placements
    name: <placementName>
  pullSecret:
    name: <pullSecretName>
  registries:
    - mirror: <mirrored-image-registry-address>
```

```

source: <image-registry-address>
- mirror: <mirrored-image-registry-address>
source: <image-registry-address>

```

在 `spec` 部分中：

- 将 `placementName` 替换为选择一组受管集群的放置名称。
- 将 `pullSecretName` 替换为用于从自定义镜像 `registry` 中拉取镜像的 `pull secret` 名称。
- 列出每个 `source` 和 `mirror registry` 的值。将 `mirrored-image-registry-address` 和 `image-registry-address` 替换为每个 `registry` 的 `mirror` 和 `source` 的值。
 - 示例 1：要将名为 `registry.redhat.io/rhacm2` 的源 `registry` 替换为 `localhost:5000/rhacm2`，并将 `registry.redhat.io/multicluster-engine` 替换为 `localhost:5000/multicluster-engine`，请使用以下示例：

```

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

```

- 示例 2：要将源镜像 `registry.redhat.io/rhacm2/registration-rhel8-operator` 替换为 `localhost:5000/rhacm2-registration-rhel8-operator`，请使用以下示例：

```

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

```

1.5.5.4.1. 导入具有 `ManagedClusterImageRegistry` 的集群

完成以下步骤以导入使用 `ManagedClusterImageRegistry` 自定义资源定义自定义的集群：

1. 在您要导入集群的命名空间中创建 `pull secret`。对于这些步骤，命名空间是 `myNamespace`。

```

$ kubectl create secret docker-registry myPullSecret \
--docker-server=<your-registry-server> \

```

```
--docker-username=<my-name> \
--docker-password=<my-password>
```

2.

在您创建的命名空间中创建一个放置。

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

注：需要 **unreachable** 容限才能使配置来选择集群。

3.

创建一个 **ManagedClusterSet** 资源，并将其绑定到命名空间。

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSet
metadata:
  name: myClusterSet

---
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
  name: myClusterSet
  namespace: myNamespace
spec:
  clusterSet: myClusterSet
```

4.

在命名空间中创建 **ManagedClusterImageRegistry** 自定义资源定义。

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

```
pullSecret:  
  name: myPullSecret  
  registry: myRegistryAddress
```

5. 从控制台导入受管集群并将其添加到受管集群集中。
6. 在标签 `open-cluster-management.io/image-registry=myNamespace.myImageRegistry` 添加到受管集群后，在受管集群中复制并运行导入命令。

1.5.6. 访问集群

要访问创建和管理的 **Red Hat OpenShift Container Platform** 集群，请完成以下步骤：

1. 在控制台中，导航到 **Infrastructure > Clusters**，再选择您创建的或想要访问的集群名称。
2. 选择 **Reveal credentials** 来查看集群的用户名和密码。记下这些值以便在登录到集群时使用。

注：**Reveal credentials** 选项不适用于导入的集群。

3. 选择 **Console URL** 以链接到集群。
4. 使用在第 3 步中找到的用户 ID 和密码登录集群。

1.5.7. 扩展受管集群

对于您创建的集群，您可以自定义并调整受管集群规格，如虚拟机大小和节点数量。查看以下选项：

- 使用 [MachinePool](#) 扩展
- 使用 [Discovery](#) 镜像将主机添加到主机清单中

1.5.7.1. 使用 [MachinePool](#) 扩展

对于使用多集群引擎 **operator** 置备的集群，会自动为您创建 **MachinePool** 资源。您可以使用 **MachinePool** 进一步自定义和调整受管集群规格，如虚拟机大小和节点数量。

- 裸机集群不支持使用 **MachinePool** 资源。
- **MachinePool** 资源是 **Hub** 集群上的 **Kubernetes** 资源，用于将 **MachineSet** 资源分组到受管集群上。
- **MachinePool** 资源统一配置一组计算机资源，包括区配置、实例类型和 **root** 存储。
- 使用 **MachinePool**，您可以手动配置所需的节点数量，或者配置受管集群中的节点自动扩展。

1.5.7.1.1. 配置自动扩展

配置自动扩展可让集群根据需要进行扩展，从而降低资源成本，在流量较低时进行缩减，并通过向上扩展以确保在资源需求较高时有足够的资源。

- 要使用控制台在 **MachinePool** 资源上启用自动扩展，请完成以下步骤：
 1. 在导航中，选择 **Infrastructure > Clusters**。
 2. 点目标集群的名称并选择 **Machine pool** 选项卡。
 3. 在 **machine pool** 页中，从目标机器池的 **Options** 菜单中选择 **Enable autoscale**。
 4. 选择机器设置副本的最小和最大数量。计算机集副本直接映射到集群中的节点。

在点 **Scale** 后，更改可能需要几分钟时间来反映控制台。您可以通过单击 **Machine pool** 选项卡中的通知中的 **View machine** 来查看扩展操作的状态。
- 要使用命令行在 **MachinePool** 资源上启用自动扩展，请完成以下步骤：

1. 输入以下命令查看您的机器池列表，将 **managed-cluster-namespace** 替换为目标受管集群的命名空间。

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

2. 输入以下命令为机器池编辑 **YAML** 文件：

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

- 将 **MachinePool-resource-name** 替换为 **MachinePool** 资源的名称。
- 将 **managed-cluster-namespace** 替换为受管集群的命名空间的名称。

3. 从 **YAML** 文件删除 **spec.replicas** 字段。

4. 在资源 **YAML** 中添加 **spec.autoscaling.minReplicas** 设置和 **spec.autoscaling.maxReplicas** 项。

5. 将最小副本数添加到 **minReplicas** 设置。

6. 将最大副本数添加到 **maxReplicas** 设置中。

7. 保存文件以提交更改。

1.5.7.1.2. 禁用自动扩展

您可以使用控制台或命令行禁用自动扩展。

- 要使用控制台禁用自动扩展，请完成以下步骤：

1. 在导航中，选择 **Infrastructure > Clusters**。

2. 点目标集群的名称并选择 **Machine pool** 选项卡。
3. 在 **machine pool** 页面中，从目标机器池的 **Options** 菜单中选择 **Disable autoscale**。
4. 选择您想要的机器集副本数量。机器集副本直接与集群中的节点映射。

在点 **Scale** 后，在控制台中显示可能需要几分钟时间。您可以点 **Machine pools** 选项卡中的通知中的 **View machine** 来查看扩展的状态。

- 要使用命令行禁用自动扩展，请完成以下步骤：

1. 输入以下命令查看您的机器池列表：

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

将 **managed-cluster-namespace** 替换为目标受管集群的命名空间。

2. 输入以下命令为机器池编辑 **YAML** 文件：

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

将 **name-of-MachinePool-resource** 替换为 **MachinePool** 资源的名称。

将 **namespace-of-managed-cluster** 替换为受管集群的命名空间的名称。

3. 从 **YAML** 文件中删除 **spec.autoscaling** 字段。
4. 将 **spec.replicas** 字段添加到资源 **YAML**。

5. 将副本数添加到 **replicas** 设置中。
6. 保存文件以提交更改。

1.5.7.1.3. 启用手动扩展

您可以从控制台和命令行手动扩展。

1.5.7.1.3.1. 使用控制台启用手动扩展

要使用控制台扩展 **MachinePool** 资源，请完成以下步骤：

1. 如果启用了 **MachinePool**，请禁用自动扩展。请参阅前面的步骤。
2. 在控制台中点 **Infrastructure > Clusters**。
3. 点目标集群的名称并选择 **Machine pool** 选项卡。
4. 在 **machine pool** 页面中，从 **Options** 菜单中为目标集群池选择 **Scale machine pool**。
5. 选择您想要的机器集副本数量。机器集副本直接与集群中的节点映射。在点 **Scale** 后，更改可能需要几分钟时间来反映控制台。您可以从 **Machine pool** 选项卡的 **notification** 中点 **View machine** 来查看扩展操作的状态。

1.5.7.1.3.2. 使用命令行启用手动扩展

要使用命令行扩展 **MachinePool** 资源，请完成以下步骤：

1. 输入以下命令查看您的机器池列表，将 **< managed-cluster-namespace >** 替换为目标受管集群命名空间的命名空间：

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

2. 输入以下命令为机器池编辑 **YAML** 文件：

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

- 将 **MachinePool-resource-name** 替换为 **MachinePool** 资源的名称。
 - 将 **managed-cluster-namespace** 替换为受管集群的命名空间的名称。
3. 从 **YAML** 文件中删除 **spec.autoscaling** 字段。
 4. 使用您想要的副本数修改 **YAML** 文件中的 **spec.replicas** 字段。
 5. 保存文件以提交更改。

1.5.8. 使用集群代理附加组件

在某些情况下，受管集群位于防火墙后面，无法由 **hub** 集群直接访问。要获取访问权限，您可以设置代理附加组件来访问受管集群的 **kube-apiserver**，以提供更安全的连接。

重要： 当 **hub** 集群或受管集群具有 **[cluster-wide proxy]**(<https://docs.openshift.com/container-platform/4.12/networking/enable-cluster-wide-proxy.html>)配置时，**cluster-proxy-addon** 无法正常工作。

需要的访问权限：**Editor**

要为 **hub** 集群和受管集群配置集群代理附加组件，请完成以下步骤：

1. 通过完成以下步骤，配置 **kubeconfig** 文件以访问受管集群 **kube-apiserver**：
 - a. 为受管集群提供有效的访问令牌。

注：您可以使用服务帐户的对应令牌。您还可以使用位于 **default** 命名空间中的

default 服务帐户。

- i. 运行以下命令导出受管集群的 **kubeconfig** 文件：

```
export KUBECONFIG=<managed-cluster-kubeconfig>
```

- ii. 在服务帐户中添加一个角色，允许它通过运行以下命令来访问 **pod**：

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

- iii. 运行以下命令来查找服务帐户令牌的 **secret**：

```
oc get secret -n default | grep <default-token>
```

将 **default-token** 替换为您的 **secret** 的名称。

- iv. 运行以下命令复制令牌：

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

将 **default-token** 替换为您的 **secret** 的名称。

- b. 在 **Red Hat Advanced Cluster Management hub** 集群中配置 **kubeconfig** 文件。

- i. 运行以下命令，在 **hub** 集群中导出当前的 **kubeconfig** 文件：

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

- ii. 使用编辑器修改 **server** 文件。本例使用 **sed**。如果您使用 **OSX**，运行 **alias sed=gsed**。

```
export TARGET_MANAGED_CLUSTER=<managed-cluster-name>
```

```
export NEW_SERVER=https://$(oc get route -n multicluster-engine cluster-proxy-
addon-user -o=jsonpath='{.spec.host}')/$TARGET_MANAGED_CLUSTER

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

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

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

iii.

输入以下命令删除原始用户凭证：

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

iv.

添加服务帐户的令牌：

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

2.

运行以下命令，列出目标受管集群的目标命名空间中的所有 **pod**：

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

将 **default** 命名空间替换为您要使用的命名空间。

3.

访问受管集群中的其他服务。当受管集群是 **Red Hat OpenShift Container Platform** 集群时，此功能可用。该服务必须使用 **service-serving-certificate** 来生成服务器证书：

•

在受管集群中，使用以下服务帐户令牌：

```
export PROMETHEUS_TOKEN=$(kubectl get secret -n openshift-monitoring $(kubectl
get serviceaccount -n openshift-monitoring prometheus-k8s -
o=jsonpath='{.secrets[0].name}') -o=jsonpath='{.data.token}' | base64 -d)
```

•

从 **hub** 集群中，运行以下命令来将证书颁发机构转换为文件：

```
oc get configmap kube-root-ca.crt -o=jsonpath='{.data.ca.crt}' > hub-ca.crt
```

4.

使用以下命令获取受管集群的 **Prometheus** 指标：

```
export SERVICE_NAMESPACE=openshift-monitoring
export SERVICE_NAME=prometheus-k8s
export SERVICE_PORT=9091
export SERVICE_PATH="api/v1/query?query=machine_cpu_sockets"
curl --cacert hub-ca.crt
$NEW_SERVER/api/v1/namespaces/$SERVICE_NAMESPACE/services/$SERVICE_NAME:$
SERVICE_PORT/proxy-service/$SERVICE_PATH -H "Authorization: Bearer
$PROMETHEUS_TOKEN"
```

1.5.9. 配置 Ansible Automation Platform 任务以在受管集群上运行

multicluster engine operator 与 **Red Hat Ansible Automation Platform** 集成，以便您可以在创建或升级集群前创建 **prehook** 和 **posthook Ansible** 作业实例。为集群销毁配置 **prehook** 和 **posthook** 作业，集群扩展操作不被支持。

需要的访问权限：集群管理员

- [先决条件](#)
- [使用控制台将 **Automation** 模板配置为在集群中运行](#)
- [创建 **Automation** 模板](#)
- [查看 **Ansible** 作业的状态](#)

1.5.9.1. 先决条件

您必须满足以下先决条件才能在集群中运行 **Automation** 模板：

- **OpenShift Container Platform 4.6 或更高版本**
- 安装 **Ansible Automation Platform Resource Operator**，将 **Ansible** 作业连接到 **Git** 订阅的生命周期。为了获得最佳结果，当使用 **Automation** 模板启动 **Ansible Automation Platform** 作业时，**Ansible Automation Platform** 作业模板在运行时应该是幂等的。您可以在 **OpenShift**

Container Platform *OperatorHub* 中找到 **Ansible Automation Platform Resource Operator**。

1.5.9.2. 使用控制台将 *Automation* 模板配置为在集群中运行

您可以在创建集群时、导入集群或创建集群时，指定要用于集群的 **Automation** 模板。

要在创建或导入集群时指定模板，请在 *Automation* 步骤中选择要应用到集群的 **Ansible** 模板。如果没有 **Automation** 模板，点 **Add automation template** 来创建。

要在创建集群时指定模板，请点击现有集群的操作菜单中的 **Update automation template**。您还可以使用 **Update automation template** 选项更新现有的自动化模板。

1.5.9.3. 创建 **Automation** 模板

要使用集群安装或升级启动 **Ansible** 作业，您必须创建一个 **Automation** 模板来指定作业何时运行。它们可以配置为在集群安装或升级之前或之后运行。

要指定在创建模板时运行 **Ansible** 模板的详情，请完成控制台中的步骤：

1. 从导航中选择 **Infrastructure > Automation**。
2. 选择适用于您的问题单的适用路径：
 - 如果要创建新模板，请单击 **Create Ansible template** 并继续第 3 步。
 - 如果要修改现有模板，请在要修改的模板的 *Options* 菜单中单击 **Edit template**，然后继续第 5 步。
3. 输入模板的唯一名称，其中包含小写字母数字字符或连字符(-)。
4. 选择您要用于新模板的凭据。要将 **Ansible** 凭证链接到 **Ansible** 模板，请完成以下步骤：

- a. 在导航中，选择 **Automation**。任何未链接到凭证的模板列表中的模板都包含可用于将模板链接到现有凭证的 **Link to credential** 图标。仅显示与模板相同的命名空间中的凭证。
 - b. 如果没有可以选择的凭证，或者您不想使用现有凭证，请从您要链接的模板的 **Options** 菜单中选择 **Edit template**。
 - c. 如果需要创建自己的凭证，点 **Add credential** 并完成 **Ansible Automation Platform 创建一个凭证** 中的步骤。
 - d. 在与模板相同的命名空间中创建凭据后，在编辑模板时，在 **Ansible Automation Platform credential** 字段中选择凭据。
5. 如果要在安装集群前启动任何 **Ansible** 作业，在 **Pre-install Automation templates** 部分中选择 **Add an Automation template**。
 6. 选择或输入 **prehook** 和 **posthook Ansible** 作业的名称，以添加到集群的安装或升级中。

注：**Automation** 模板名称必须与 **Ansible Automation Platform** 上的 **Ansible** 作业的名称匹配。
 7. 如有必要，拖动 **Ansible** 作业以更改顺序。
 8. 对于您要在 **Post-install Automation templates** 部分、**Pre-upgrade Automation templates** 和 **Post-upgrade Automation templates** 部分安装集群后启动的任何自动化模板，重复步骤 5 - 7。

您的 **Ansible** 模板已配置为在集群中运行，在指定操作发生时指定此模板。

1.5.9.4. 查看 **Ansible** 作业的状态

您可以查看正在运行的 **Ansible** 作业的状态，以确保它启动并在成功运行。要查看正在运行的 **Ansible** 作业的当前状态，请完成以下步骤：

1. 在菜单中，选择 **Infrastructure > Clusters** 以访问 **Clusters** 页面。
2. 选择集群名称来查看其详情。
3. 在集群信息上查看 **Ansible** 作业最后一次运行的状态。该条目显示以下状态之一：
 - 当安装 **prehook** 或 **posthook** 任务失败时，集群状态会显示 **Failed**。
 - 当升级 **prehook** 或 **posthook** 任务失败时，会在 **Distribution** 字段中显示升级失败的警告信息。

提示：如果集群 **prehook** 或 **posthook** 失败，您可以从 **Clusters** 页面重试升级。

1.5.10. ClusterClaims

ClusterClaim 是受管集群中的一个集群范围的自定义资源定义（CRD）。**ClusterClaim** 代表受管集群声明的一个信息片段。您可以使用 **ClusterClaim** 来降级目标集群上资源的放置。

以下示例显示了 **YAML** 文件中标识的 **ClusterClaim**：

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: id.openshift.io
spec:
  value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
```

下表显示了在多集群引擎 **Operator** 管理的集群中定义的 **ClusterClaims**：

声明名称	保留	可变	描述
id.k8s.io	true	false	在上游社区定义的 ClusterID
kubeversion.open-cluster-management.io	true	true	Kubernetes 版本

声明名称	保留	可变	描述
platform.open-cluster-management.io	true	false	运行受管集群的平台，如 AWS、GCE 和 Equinix Metal
product.open-cluster-management.io	true	false	产品名称，如 OpenShift、Anthos、EKS 和 GKE
id.openshift.io	false	false	OpenShift Container Platform 外部 ID，它仅适用于 OpenShift Container Platform 集群
consoleurl.openshift.io	false	true	管理控制台的 URL，仅适用于 OpenShift Container Platform 集群
version.openshift.io	false	true	OpenShift Container Platform 版本，它仅适用于 OpenShift Container Platform 集群

如果在受管集群中删除或更新之前的声明，它们会自动恢复或回滚到上一版本。

在受管集群加入 **hub** 后，在受管集群上创建的 **ClusterClaims** 与 **hub** 上的 **ManagedCluster** 资源的状态同步。带有 **ClusterClaims** 的受管集群可能类似以下示例：

```

apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  labels:
    cloud: Amazon
    clusterID: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
    installer.name: multiclusterhub
    installer.namespace: open-cluster-management
    name: cluster1
    vendor: OpenShift
    name: cluster1
  spec:
    hubAcceptsClient: true
    leaseDurationSeconds: 60
  status:
    allocatable:
      cpu: '15'
      memory: 65257Mi
    capacity:
      cpu: '18'

```

```

memory: 72001 Mi
clusterClaims:
  - name: id.k8s.io
    value: cluster1
  - name: kubeversion.open-cluster-management.io
    value: v1.18.3+6c42de8
  - name: platform.open-cluster-management.io
    value: AWS
  - name: product.open-cluster-management.io
    value: OpenShift
  - name: id.openshift.io
    value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
  - name: consoleurl.openshift.io
    value: 'https://console-openshift-console.apps.xxxx.dev04.red-chesterfield.com'
  - name: version.openshift.io
    value: '4.5'
conditions:
  - lastTransitionTime: '2020-10-26T07:08:49Z'
    message: Accepted by hub cluster admin
    reason: HubClusterAdminAccepted
    status: 'True'
    type: HubAcceptedManagedCluster
  - lastTransitionTime: '2020-10-26T07:09:18Z'
    message: Managed cluster joined
    reason: ManagedClusterJoined
    status: 'True'
    type: ManagedClusterJoined
  - lastTransitionTime: '2020-10-30T07:20:20Z'
    message: Managed cluster is available
    reason: ManagedClusterAvailable
    status: 'True'
    type: ManagedClusterConditionAvailable
version:
  kubernetes: v1.18.3+6c42de8

```

1.5.10.1. 列出现有 ClusterClaims

您可以使用 `kubectl` 命令列出应用到受管集群的 `ClusterClaims`。当您要将 `ClusterClaim` 与错误消息进行比较时，这很有用。

备注：您需要具有 `clusterclaims.cluster.open-cluster-management.io` 资源的 `list` 权限。

运行以下命令列出受管集群中的所有现有 `ClusterClaims`：

```
kubectl get clusterclaims.cluster.open-cluster-management.io
```

1.5.10.2. 创建自定义 ClusterClaims

您可以使用受管集群上的自定义名称创建 **ClusterClaims**，这样可更轻松地识别它们。自定义 **ClusterClaims** 会与 **hub** 集群上的 **ManagedCluster** 资源进行同步。以下内容显示了自定义 **ClusterClaim** 的定义示例：

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: <custom_claim_name>
spec:
  value: <custom_claim_value>
```

字段 **spec.value** 的最大长度为 **1024**。创建 **ClusterClaim** 需要资源 **clusterclaims.cluster.open-cluster-management.io** 的 **create** 权限。

1.5.11. *ManagedClusterSets*

ManagedClusterSet 是一个受管集群的组。受管集群集可帮助您管理对所有受管集群的访问。您还可以创建一个 **ManagedClusterSetBinding** 资源，将 **ManagedClusterSet** 资源绑定到命名空间。

每个集群都必须是受管集群集的成员。安装 **hub** 集群时，会创建一个名为 **default** 的 **ManagedClusterSet** 资源。所有没有分配给受管集群集的集群会自动分配给默认的受管集群集。您不能删除或更新默认受管集群集。

继续阅读以了解更多有关如何创建和管理受管集群集的信息：

- [创建 *ManagedClusterSet*](#)
- [为 *ManagedClusterSets* 分配 RBAC 权限](#)
- [创建 *ManagedClusterSetBinding* 资源](#)
- [从 *ManagedClusterSet* 中删除集群](#)

1.5.11.1. 创建 *ManagedClusterSet*

您可以在受管集群集中将受管集群分组在一起，以限制受管集群的用户访问权限。

需要的访问权限：集群管理员

ManagedClusterSet 是一个集群范围的资源，因此您必须在创建 **ManagedClusterSet** 的集群中具有集群管理权限。受管集群不能包含在多个 **ManagedClusterSet** 中。您可以通过 **multicluster engine operator** 控制台或 **CLI** 创建受管集群集。

注：没有添加到受管集群集中的集群池不会添加到默认的 **ManagedClusterSet** 资源中。从集群池中声明集群后，集群将添加到默认的 **ManagedClusterSet** 中。

在创建受管集群时，会自动创建以下内容来简化管理：

- 名为 **global** 的 **ManagedClusterSet**。
- 名为 **open-cluster-management-global-set** 的命名空间。
- 名为 **global** 的 **ManagedClusterSetBinding**，将全局 **ManagedClusterSet** 绑定到 **open-cluster-management-global-set** 命名空间。

重要：您无法删除、更新或编辑全局受管集群集。全局受管集群集包含所有受管集群。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
  name: global
  namespace: open-cluster-management-global-set
spec:
  clusterSet: global
```

1.5.11.1.1. 使用 CLI 创建 ManagedClusterSet

将受管集群集的以下定义添加到 **YAML** 文件中，以使用 **CLI** 创建受管集群集：

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSet
metadata:
  name: <cluster_set>
```

将 `<cluster_set >` 替换为受管集群的名称。

1.5.11.1.2. 将集群添加到 *ManagedClusterSet*

创建 **ManagedClusterSet** 后，您可以按照控制台中的说明或使用 **CLI** 将集群添加到受管集群集中。

1.5.11.1.3. 使用 **CLI** 将集群添加到 *ManagedClusterSet*

完成以下步骤，使用 **CLI** 将集群添加到受管集群集中：

1. 确保有一个 **RBAC ClusterRole** 条目，供您在 `managedclustersets/join` 的虚拟子资源中创建。

注：如果没有此权限，您无法将受管集群分配给 **ManagedClusterSet**。如果此条目不存在，将其添加到 **YAML** 文件中。请参见以下示例：

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clusterrole1
rules:
- apiGroups: ["cluster.open-cluster-management.io"]
  resources: ["managedclustersets/join"]
  resourceNames: ["<cluster_set>"]
  verbs: ["create"]
```

将 `<cluster_set >` 替换为 **ManagedClusterSet** 的名称。

注：如果要受管集群从一个 **ManagedClusterSet** 移到另一个，则必须在两个受管集群集中都有该权限。

2. 在 **YAML** 文件中找到受管集群的定义。请参见以下示例定义：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: <cluster_name>
spec:
  hubAcceptsClient: true
```

3.

添加 `cluster.open-cluster-management.io/clusterset` 参数，并指定 `ManagedClusterSet` 的名称。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: <cluster_name>
  labels:
    cluster.open-cluster-management.io/clusterset: <cluster_set>
spec:
  hubAcceptsClient: true
```

1.5.11.2. 为 `ManagedClusterSet` 分配 RBAC 权限

您可以将用户或组分配给由 `hub` 集群上配置的身份提供程序提供的集群集合。

需要的访问权限：集群管理员

下表列出了三个 `ManagedClusterSet` API RBAC 权限级别：

集群集	访问权限	创建权限
admin	对分配给受管集群集的所有集群和集群池资源具有完全访问权限。	创建集群、导入集群和创建集群池的权限。创建时，必须将权限分配给受管集群集。
bind	通过创建一个 ManagedClusterSetBinding ，将集群集绑定到命名空间的权限。用户或组还必须具有在目标命名空间中创建 ManagedClusterSetBinding 的权限。对分配给受管集群集的所有集群和集群池资源只读权限。	没有创建集群、导入集群或创建集群池的权限。
view	对分配给受管集群集的所有集群和集群池资源进行只读权限。	没有创建集群、导入集群或创建集群池的权限。

注：您不能为全局集群集应用 **Cluster set admin** 权限。

完成以下步骤，从控制台将用户或组分配给受管集群集：

1. 在 **OpenShift Container Platform** 控制台中进入到 **Infrastructure > Clusters**。
2. 选择 **Cluster sets** 选项卡。
3. 选择您的目标集群集。
4. 选择 **Access management** 选项卡。
5. 选择 **Add user or group**。
6. 搜索，然后选择您要提供访问权限的用户和组。
7. 选择 **Cluster set admin** 或 **Cluster set view** 角色，赋予所选用户或用户组。如需更多信息，请参阅 [多集群引擎 operator 基于角色的访问控制](#) 中的角色概述。
8. 选择 **Add** 以提交更改。

表中会显示您的用户或组。可能需要几秒钟后，分配到所有受管集群设置的资源的权限才会被传播到您的用户或组。

如需放置信息，请参阅使用 [ManagedClusterSet with Placement](#)。

1.5.11.3. 创建 *ManagedClusterSetBinding* 资源

ManagedClusterSetBinding 资源将 **ManagedClusterSet** 资源绑定到命名空间。在同一命名空间中创建的应用程序和策略只能访问包含在绑定受管集群集资源中的集群。

命名空间的访问权限会自动应用到绑定到该命名空间的受管集群集。如果您有对该命名空间的访问权限，则会自动访问绑定到该命名空间的任何受管集群集的权限。如果您只具有访问受管集群集的权限，则

不会自动具有访问命名空间中其他受管集群集的权限。

您可以使用控制台或命令行创建受管集群集绑定。

1.5.11.3.1. 使用控制台创建 *ManagedClusterSetBinding*

完成以下步骤，使用控制台创建 *ManagedClusterSetBinding*：

1. 在 **OpenShift Container Platform** 控制台中进入到 **Infrastructure > Clusters** 并选择 **Cluster set** 选项卡。
2. 选择您要为其创建绑定的集群集的名称。
3. 导航到 **Actions > Edit namespace bindings**。
4. 在 **Edit namespace bindings** 页面中，从下拉菜单中选择您要将其绑定到的命名空间。

1.5.11.3.2. 使用 CLI 创建 *ManagedClusterSetBinding*

完成以下步骤，使用 CLI 创建 *ManagedClusterSetBinding*：

1. 在 **YAML** 文件中创建 *ManagedClusterSetBinding* 资源。

注：当您创建受管集群集绑定时，受管集群集绑定的名称必须与要绑定的受管集群集的名称匹配。您的 *ManagedClusterSetBinding* 资源可能类似以下信息：

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
  namespace: <namespace>
  name: <cluster_name>
spec:
  clusterSet: <cluster_set>
```

2. 确保目标受管集群集有绑定权限。查看以下 *ClusterRole* 资源示例，其中包含允许用户绑定

到 `< cluster_set >` 的规则：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <clusterrole>
rules:
- apiGroups: ["cluster.open-cluster-management.io"]
  resources: ["managedclustersets/bind"]
  resourceNames: ["<cluster_set>"]
  verbs: ["create"]
```

1.5.11.4. 使用污点和容限放置受管集群

您可以使用污点和容限控制受管集群或受管集群集的放置。污点和容限提供了一种防止为特定放置选择受管集群的方法。如果要阻止某些受管集群包含在某些放置中，这个控制会很有用。您可以向受管集群添加污点，并为放置添加容限。如果污点和容限不匹配，则不会为该放置选择受管集群。

1.5.11.4.1. 将污点添加到受管集群

污点在受管集群的属性中指定，并允许放置来重新放置受管集群或一组受管集群。您可以通过输入类似以下示例的命令，为受管集群添加污点：

```
oc taint ManagedCluster <managed_cluster_name> key=value:NoSelect
```

污点的规格包括以下字段：

- **必需键** - 应用到集群的污点键。这个值必须与受管集群的容限中的值匹配，以满足添加到该放置的条件。您可以确定这个值。例如，这个值可以是 **bar** 或 **foo.example.com/bar**。
- **可选值** - 污点键的污点值。这个值必须与受管集群的容限中的值匹配，以满足添加到该放置的条件。例如，这个值可以是 **value**。
- **必需效果** - 污点对不容许污点的放置效果，或者在污点和放置容限不匹配时发生什么。**effect** 的值必须是以下值之一：
 - **NoSelect** - 除非容许这个污点，否则不允许放置来选择集群。如果在设置污点前放置选择了集群，则会从放置决定中移除集群。
 -

NoSelectIfNew - 如果是新集群，调度程序就无法选择该集群。只有容许污点且已在其集群决策中拥有集群，放置才可以选择集群。

- 必需 **TimeAdded** - 添加污点的时间。这个值会自动设置。

1.5.11.4.2. 识别内置污点以反映受管集群的状态

当无法访问受管集群时，您不希望集群添加到放置中。以下污点会自动添加到无法访问的受管集群：

- **cluster.open-cluster-management.io/unavailable** - 当集群有 **ManagedClusterConditionAvailable** 条件为 **False** 时，这个污点被添加到受管集群。污点具有 **NoSelect** 和空值的效果，以防止调度不可用的集群。以下内容中提供了此污点的示例：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
taints:
  - effect: NoSelect
    key: cluster.open-cluster-management.io/unavailable
    timeAdded: '2022-02-21T08:11:54Z'
```

- **cluster.open-cluster-management.io/unreachable** - 当 **ManagedClusterConditionAvailable** 条件的状态为 **Unknown** 或没有条件时，此污点被添加到受管集群。污点对 **NoSelect** 和一个空值的影响，以防止调度无法访问的集群。以下内容中提供了此污点的示例：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
taints:
  - effect: NoSelect
    key: cluster.open-cluster-management.io/unreachable
    timeAdded: '2022-02-21T08:11:06Z'
```

1.5.11.4.3. 为放置添加容限

容限应用到放置，并允许放置没有与放置容限匹配的污点的受管集群。容限的规格包括以下字段：

- 可选 键 - 密钥与 **taint** 键匹配以允许放置。
- 可选 值 - 容限中的值必须与容限的污点值匹配，以允许放置。
- 可选 **Operator** - **Operator** 代表键和值之间的关系。有效的操作符是 **equal** 和 **exists**。默认值为 **equal**。当键相同时，容限与污点匹配，影响相同，操作符是以下值之一：
 - **equal** - 操作符 **equal**，值在污点和容忍度中是相同的。
 - **exists** - 值的通配符，因此放置可以容限特定类别的所有污点。
- 可选 效果 - 要匹配的污点效果。当留空时，它将匹配所有污点效果。指定后允许的值为 **NoSelect** 或 **NoSelectIfNew**。
- 可选 **TolerationSeconds** - 将受管集群移至新放置前容许污点的时间长度（以秒为单位）。如果 **effect** 值不是 **NoSelect** 或 **PreferNoSelect**，会忽略此字段。默认值为 **nil**，这表示没有时间限制。**TolerationSeconds** 的开始时间自动列为污点中的 **TimeAdded** 值，而不是在集群调度时间或者 **TolerationSeconds** 添加时间。

以下示例演示了如何配置容许具有污点的集群的容限：

- 受管集群中的污点，例如：

```

apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: gpu
      value: "true"
      timeAdded: '2022-02-21T08:11:06Z'

```

- 允许容许污点的放置上的容限

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal

```

在定义了容限示例后，放置可以选择 **cluster1**，因为 **key: gpu** 和 **value: "true"** 匹配。

注：受管集群不能保证放在包含污点容限的放置上。如果其他放置包含相同的容限，受管集群可能会放置到其中一个放置上。

1.5.11.4.4. 指定临时容限

TolerationSeconds 的值指定容限容许污点的时间期限。当受管集群离线时，这个临时容限会很有用，您可以将在此集群中部署的应用程序传送到另一个受管集群中，以便容忍的时间。

例如，具有以下污点的受管集群将无法访问：

```

apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: cluster.open-cluster-management.io/unreachable
      timeAdded: '2022-02-21T08:11:06Z'

```

如果您使用 **TolerationSeconds** 的值定义放置，如下例所示，工作负载在 5 分钟后传输到另一个可用的受管集群。

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: demo4
  namespace: demo1
spec:
  tolerations:

```

- key: cluster.open-cluster-management.io/unreachable
operator: Exists
tolerationSeconds: 300

受管集群在无法访问 5 分钟后，应用程序会被移到另一个受管集群。

1.5.11.5. 从 *ManagedClusterSet* 中删除受管集群

您可能希望从受管集群集中删除受管集群，将其移到不同的受管集群集，或者从集合的管理设置中删除。您可以使用控制台或 CLI 从受管集群集中删除受管集群。

注：每个受管集群都必须分配到一个受管集群集。如果您从 *ManagedClusterSet* 中删除受管集群，且没有将其分配给不同的 *ManagedClusterSet*，集群会自动添加到默认受管集群集中。

1.5.11.5.1. 使用控制台从 *ManagedClusterSet* 中删除集群

完成以下步骤，使用控制台从受管集群集中删除集群：

1. 点 **Infrastructure > Clusters**，并确保选择了 *Cluster set* 选项卡。
2. 选择您要从受管集群集中删除的集群集的名称，以查看集群设置详情。
3. 选择 **Actions > Manage resource assignments**。
4. 在 *Manage resource assignments* 页面上，删除您要从集群集中删除的资源的复选框。

此步骤删除已是集群集成员的资源。您可以通过查看受管集群的详情来查看资源是否已是集群集的成员。

注：如果要受管集群从一个受管集群集移到另一个受管集群，则必须在两个受管集群集中都所需的 RBAC 权限。

1.5.11.5.2. 使用 CLI 从 *ManagedClusterSet* 中删除集群

要使用命令行从受管集群集中删除集群，请完成以下步骤：

1. 运行以下命令在受管集群集中显示受管集群列表：

```
oc get managedclusters -l cluster.open-cluster-management.io/clusterSet=<cluster_set>
```

将 **cluster_set** 替换为受管集群集的名称。

2. 找到您要删除的集群条目。
3. 从您要删除的集群的 **YAML** 条目中删除标签。参阅以下标签代码示例：

```
labels:
  cluster.open-cluster-management.io/clusterSet: clusterSet1
```

注：如果要将在受管集群从一个集群设置为另一个集群，则必须在两个受管集群集中都有所需的 **RBAC** 权限。

1.5.12. 在放置中使用 ManagedClusterSet

Placement 资源是一个命名空间范围的资源，它定义了一个规则来从 **ManagedClusterSets** 中选择 **ManagedClusters** 集合，它们绑定到放置命名空间。

需要的访问权限：**Cluster administrator**, **Cluster set administrator**

1.5.12.1. 放置概述

参阅以下有关使用受管集群放置的信息：

- **Kubernetes** 集群在 **hub** 集群中注册，作为集群范围的 **ManagedClusters**。
- **ManagedClusters** 被组织到集群范围的 **ManagedClusterSets** 中。
- **ManagedClusterSets** 与工作负载命名空间绑定。

- 命名空间范围的放置指定 **ManagedClusterSet** 的一个部分，用于选择潜在 **ManagedClusters** 的工作集合。
- **placements** 使用标签和声明选择器从受管集群集过滤 **ManagedClusters**。
- **ManagedClusters** 的放置可以使用污点和容限控制。如需更多信息，[请参阅使用污点和容限放置受管集群](#)。
- 放置 根据要求对集群进行排名，并从中选择集群的子集。

备注：

- 您需要通过在该命名空间中创建一个 **ManagedClusterSetBinding** 来最少将一个 **ManagedClusterSet** 绑定到一个命名空间。
- 您需要基于角色的访问权限，访问 **managedclustersets/bind** 的虚拟子资源上的 **CREATE**。

请参阅 [Placements API](#) 以了解有关 **API** 的详情。

1.5.12.2. placement LabelSelector 和 ClaimSelector

- 您可以使用 **labelSelector** 选择 **ManagedClusters**。请参阅以下示例，其中 **labelSelector** 仅与带有标签 **vendor: OpenShift** 的集群匹配：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
```

-

您可以使用 **claimSelector** 选择 **ManagedClusters**。请参阅以下示例，其中 **claimSelector** 仅与带有 **us-west-1** 的 **region.open-cluster-management.io** 的集群匹配：

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
      claimSelector:
        matchExpressions:
          - key: region.open-cluster-management.io
            operator: In
            values:
              - us-west-1
  
```

您可以从特定的 **clusterSets** 中选择 **ManagedClusters**。请参阅以下示例，其中 **claimSelector** 仅与 **clusterSets: clusterset1 clusterset2** 匹配：

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  clusterSets:
    - clusterset1
    - clusterset2
  predicates:
    - requiredClusterSelector:
      claimSelector:
        matchExpressions:
          - key: region.open-cluster-management.io
            operator: In
            values:
              - us-west-1
  
```

选择所需的 **ManagedClusters** 数量。请参阅以下示例，其中 **numberOfClusters** 为 **3**：

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 3
  predicates:
    - requiredClusterSelector:
  
```

```

labelSelector:
  matchLabels:
    vendor: OpenShift
claimSelector:
  matchExpressions:
    - key: region.open-cluster-management.io
      operator: In
      values:
        - us-west-1

```

1.5.12.3. 放置容限

- 选择具有匹配污点的 **ManagedClusters**。

假设受管集群有一个污点，如下例所示：

```

apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: gpu
      value: "true"
      timeAdded: '2022-02-21T08:11:06Z'

```

默认情况下，除非定义容限，否则放置无法选择这个集群，如下例所示：

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal

```

- 在指定时间段内选择具有匹配污点的 **ManagedClusters**。

TolerationSeconds 代表容限容许污点的时间长度，在受管集群离线时很有用，用户可以在容许的时间后将部署的应用程序转移到另一个可用的受管集群。**TolerationSeconds** 可以自动将在集群中部署的应

用程序传送到另一个受管集群，这些集群在指定时长后离线。

例如，由以下示例内容定义的受管集群将无法访问：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: cluster.open-cluster-management.io/unreachable
      timeAdded: '2022-02-21T08:11:06Z'
```

如果您使用 **TolerationSeconds** 定义放置，如以下示例所示，工作负载将在 **5** 分钟后传输到另一个可用的受管集群。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  tolerations:
    - key: cluster.open-cluster-management.io/unreachable
      operator: Exists
      tolerationSeconds: 300
```

1.5.12.4. placement PrioritizerPolicy

- 选择具有最大可分配内存的集群。

注：与 Kubernetes **Node Allocatable** 类似，“可分配”这里定义为每个集群中 **pod** 可用的计算资源数量。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
```

```

configurations:
  - scoreCoordinate:
      builtIn: ResourceAllocatableMemory

```

- 选择具有最大可分配 **CPU** 和内存的集群，并区分资源更改。

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableCPU
          weight: 2
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
          weight: 2

```

- 选择具有最大 **addOn** 分数 **CPU** 比例的两个集群，并固定放置决策。

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 2
  prioritizerPolicy:
    mode: Exact
    configurations:
      - scoreCoordinate:
          builtIn: Steady
          weight: 3
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: cpuratio

```

1.5.12.5. 放置决定

将创建一个或多个带有标签 `cluster.open-cluster-management.io/placement={placement name}` 的 **PlacementDecisions** 来代表由一个 **Placement** 选择的 **ManagedClusters**。

如果选择了 **ManagedCluster** 并添加到 **PlacementDecision** 中，消耗此放置的组件可能会在这个 **ManagedCluster** 上应用工作负载。当 **ManagedCluster** 不再被选择并从 **PlacementDecisions** 中删除后，应删除在此 **ManagedCluster** 上应用的工作负载。

请参阅 [PlacementDecisions API](#) 以了解有关 **API** 的更多详细信息。

请参阅以下 **PlacementDecision** 示例：

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/placement: placement1
  name: placement1-kbc7q
  namespace: ns1
  ownerReferences:
    - apiVersion: cluster.open-cluster-management.io/v1beta1
      blockOwnerDeletion: true
      controller: true
      kind: Placement
      name: placement1
      uid: 05441cf6-2543-4ecc-8389-1079b42fe63e
status:
  decisions:
    - clusterName: cluster1
      reason: ""
    - clusterName: cluster2
      reason: ""
    - clusterName: cluster3
      reason: ""

```

1.5.12.6. 附加组件状态

您可能希望根据部署在其上的附加组件的状态，为放置选择受管集群。例如，只有在集群中启用了特定的附加组件时，才会为放置选择受管集群。

在创建放置时，指定附加组件的标签及其状态。如果集群中启用了附加组件，则会自动在 **ManagedCluster** 资源上创建一个标签。如果禁用了附加组件，则会自动删除该标签。

每个附加组件都由一个标签表示，格式为 **feature.open-cluster-management.io/addon-
<addon_name>=<status_of_addon>**。

使用要在要选择的受管集群中启用的附加组件名称替换 **addon_name**。

将 **status_of_addon** 替换为在选择集群时该附加组件应具有的状态。**status_of_addon** 的可能值位于以下列表中：

- **available** : 附加组件已启用并可用。
- **unhealthy** : 附加组件已启用，但租期不会持续更新。
- **unreachable** : 附加组件已启用，但没有为其找到租用。也可以在受管集群离线时导致这个问题。

例如，可用的 **application-manager** 附加组件由受管集群中的标签表示：

```
feature.open-cluster-management.io/addon-application-manager: available
```

请参阅以下基于附加组件及其状态创建放置的示例：

- 您可以通过添加以下 **YAML** 内容来创建放置，其中包含启用了 **application-manager** 的所有受管集群：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: ns1
spec:
  predicates:
  - requiredClusterSelector:
    labelSelector:
      matchExpressions:
      - key: feature.open-cluster-management.io/addon-application-manager
        operator: Exists
```

- 您可以通过添加以下 **YAML** 内容来创建放置，其中包含启用了 **application-manager** 且具有 **available** 状态的所有受管集群：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
```

```

kind: Placement
metadata:
  name: placement2
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
      labelSelector:
        matchLabels:
          "feature.open-cluster-management.io/addon-application-manager": "available"

```

- 您可以通过添加以下 **YAML** 内容来创建包含禁用 **application-manager** 的所有受管集群的放置：

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement3
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
      labelSelector:
        matchExpressions:
          - key: feature.open-cluster-management.io/addon-application-manager
            operator: DoesNotExist

```

1.5.13. 管理集群池（技术预览）

集群池提供对按需和规模配置的 **Red Hat OpenShift Container Platform** 集群的快速、经济的访问。集群池在 **Amazon Web Services**、**Google Cloud Platform** 或 **Microsoft Azure** 上置备可配置且可扩展的 **OpenShift Container Platform** 集群，在需要时可以声明。在为开发、持续集成和生产环境提供或替换集群环境时，它们特别有用。您可以指定多个集群来保持运行，以便可以立即声明它们，而集群的剩余部分将保留在休眠状态，以便在几分钟后恢复并声明。

ClusterClaim 资源用于从集群池中签出集群。创建集群声明时，池会为其分配一个正在运行的集群。如果没有正在运行的集群可用，则会恢复休眠集群来提供集群或新的集群。集群池自动创建新集群，并恢复休眠集群来维护池中的指定大小和可用集群的数量。

- [创建集群池](#)
- [从集群池中声明集群](#)

- [更新集群池发行镜像](#)
- [扩展集群池](#)
- [销毁一个集群池](#)

创建集群池的过程与创建集群的步骤类似。在集群池中创建的集群不会供立即使用。

1.5.13.1. 创建集群池

创建集群池的过程与创建集群的步骤类似。在集群池中创建的集群不会供立即使用。

需要的访问权限：**Administrator**

1.5.13.1.1. 先决条件

在创建集群池前，请查看以下先决条件：

- 您需要部署多集群引擎 **operator hub** 集群。
- 您需要对多集群引擎 **operator hub** 集群的互联网访问，以便它在供应商环境中创建 **Kubernetes** 集群。
- 您需要一个 **AWS**、**GCP** 或 **Microsoft Azure** 供应商凭证。如需更多信息，请参阅[管理凭证概述](#)。
- 您需要供应商环境中配置的域。有关如何配置域的说明，请参阅您的供应商文档。
- 您需要供应商登录凭证。

●

您需要 **OpenShift Container Platform** 镜像 **pull secret**。请参阅[使用镜像 pull secret](#)。

注：使用此流程添加集群池，以便在从池中声明集群时自动导入由 **multicluster engine operator** 管理的集群。如果要创建一个没有自动导入使用集群声明的集群池，请将以下注解添加到 **clusterClaim** 资源中：

```
kind: ClusterClaim
metadata:
  annotations:
    cluster.open-cluster-management.io/createmanageredcluster: "false"
```

"false" 必须用引号括起来，表示它是一个字符串。

1.5.13.1.2. 创建集群池

要创建集群池，在导航菜单中选择 **Infrastructure > Clusters**。 **Cluster pool** 选项卡列出您可以访问的集群池。选择 **Create cluster pool** 并完成控制台中的步骤。

如果您没有要用于集群池的基础架构凭证，可以通过选择 **Add credential** 来创建一个。

您可以从列表中选择现有命名空间，或者键入要创建新命名空间的名称。集群池不必与集群位于同一个命名空间中。

如果您希望集群池的 **RBAC** 角色共享现有集群集合的角色分配，请选择集群设置名称。只有创建集群池时，才能设置集群池中的集群设置。您无法在创建集群池后更改集群池或集群池中集群集的关联。从集群池中声明的任何集群都会自动添加到与集群池相同的集群集合中。

注：如果没有 **cluster admin** 权限，则必须选择一个集群集。如果您在此情形中没有包含集群集的名称，则创建集群集的请求将被拒绝，并带有禁止的错误。如果没有集群集可供选择，请联络您的集群管理员来创建集群集，并为您提供 **clusterset admin** 权限。

集群池大小 指定您要在集群池中置备的集群数量，而集群池运行计数指定池正在运行的集群数量，并准备好立即使用。

此过程与创建集群的步骤非常相似。

有关您的供应商所需信息的详情，请查看以下信息：

- [在 Amazon Web Services 上创建集群](#)
- [在 Google Cloud Platform 上创建集群](#)
- [在 Microsoft Azure 上创建集群](#)

1.5.13.2. 从集群池中声明集群

ClusterClaim 资源用于从集群池中签出集群。当集群正在运行并位于集群池中时，会出现一个声明。集群池自动在集群池中创建新运行和休眠集群，以维护为集群池指定的要求。

注：当从集群池中声明的集群不再需要并销毁时，资源会被删除。集群不会返回到集群池。

需要的访问权限：**Administrator**

1.5.13.2.1. 前提条件

在从集群池中声明集群前，您必须有以下可用：

具有或没有可用集群的集群池。如果集群池中存在可用的集群，则会声明可用的集群。如果集群池中
没有可用的集群，则会创建一个集群来满足这个声明。如需有关如何创建集群池的信息，请参阅[创建集群池](#)。

1.5.13.2.2. 从集群池中声明集群

在创建集群声明时，您可以从集群池请求新集群。当集群可用时，从池中签出集群。声明的集群自动
导入为其中一个受管集群，除非您禁用了自动导入。

完成以下步骤以声明集群：

1. 在导航菜单中点 **Infrastructure > Clusters**，然后选择 **Cluster pool** 选项卡。
2. 从中查找您要声明集群的集群池的名称，然后选择 **Claim cluster**。

如果集群可用，它将被声明并立即出现在 **Managed cluster** 选项卡中。如果没有可用的集群，可能需要几分钟时间来恢复休眠集群或置备新集群。在这个时间中，声明状态为 **pending**。扩展集群池，以查看或删除待处理的声明。

当它在集群池中，声明的集群会保留作为与它关联的集群集的成员。在声明集群时，您无法更改声明的集群集合。

注：对于从集群池声明的现有集群，对云供应商凭证的 **pull secret**、**SSH** 密钥或基域的更改不会反映，因为它们已使用原始凭证置备。您不能使用控制台编辑集群池信息，但您可以使用 **CLI** 接口更新其信息来更新它。您还可以使用包含更新信息的凭证创建新集群池。在新池中创建的集群使用新凭证中提供的设置。

1.5.13.3. 更新集群池发行镜像

当集群中的集群保留在休眠状态一段时间时，集群的 **Red Hat OpenShift Container Platform** 发行镜像可能会变为 **backlevel**。如果发生这种情况，您可以升级集群池中集群的发行镜像版本。

需要的访问权限：**Edit**

完成以下步骤，为集群池中的集群更新 **OpenShift Container Platform** 发行镜像：

注：此步骤不会从集群池中已声明的集群池中更新集群。完成此步骤后，对发行镜像的更新仅适用于与集群池相关的以下集群：

- 使用此流程更新发行镜像后，由集群池创建的集群。
- 在集群池中休眠的集群。使用旧发行镜像的现有休眠集群将被销毁，新集群使用新的发行镜像替换它们。

1. 在导航菜单中点 **Infrastructure > Clusters**。
2. 选择 **Cluster pools** 选项卡。
3. 在 **Cluster pool** 表中找到您要更新的集群池的名称。
4. 点表中的 **Cluster pools** 的 **Options** 菜单，然后选择 **Update release image**。
5. 从这个集群池中选择一个新的发行镜像，用于将来的集群创建。

集群池发行镜像已更新。

提示：您可以通过选择一个操作来更新多个集群池的发行镜像，方法是选择每个集群池的复选框，并使用 **Actions** 菜单来更新所选集群池的发行镜像。

1.5.13.4. 扩展集群池（技术预览）

您可以通过增加或减少集群池大小中的集群数量来更改集群池中的集群数量。

需要的访问权限：集群管理员

完成以下步骤以更改集群池中的集群数量：

1. 在导航菜单中点 **Infrastructure > Clusters**。
2. 选择 **Cluster pools** 选项卡。
3. 在您要更改的集群池的 **Options** 菜单中，选择 **Scale cluster pool**。

4. 更改池大小的值。
5. 另外，您还可以更新正在运行的集群数量，以便在声明它们时立即可用的集群数量。

集群池已扩展，以反映您的新值。

1.5.13.5. 销毁一个集群池

如果您创建了集群池，并确定您不再需要它，您可以销毁集群池。当您销毁集群池时，所有未声明的休眠集群都会被销毁，并释放其资源。

需要的访问权限：集群管理员

要销毁集群池，请完成以下步骤：

1. 在导航菜单中点 **Infrastructure > Clusters**。
2. 选择 **Cluster pools** 选项卡。
3. 在您要删除的集群池的 **Options** 菜单中，选择 **Destroy cluster pool**。集群池中的任何未声明的集群都会被销毁。可能需要一些时间才能删除所有资源，集群池会在控制台中可见，直到所有资源都被删除为止。

包含 **ClusterPool** 的命名空间不会被删除。删除命名空间将销毁 **ClusterPool** 声明的任何集群，因为这些集群的 **ClusterClaim** 资源会在同一命名空间中创建。

提示：您可以通过选择一个操作来销毁多个集群池，方法是选择每个集群池的复选框，并使用 **Actions** 菜单销毁所选集群池。

1.5.14. 启用 **ManagedServiceAccount** 附加组件（技术预览）

安装多集群引擎 **operator** 时，**ManagedServiceAccount** 附加组件会被默认禁用。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](#)。

1.5.14.1. 先决条件

- **Red Hat OpenShift Container Platform** 版本 **4.11** 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 **multicluster engine operator**。

1.5.14.2. 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
```

1.5.15. 集群生命周期高级配置

您可以在安装过程中或安装后配置一些集群设置。

1.5.15.1. 自定义 API 服务器证书

受管集群通过与 **OpenShift Kube API** 服务器外部负载均衡器相互连接与 **hub** 集群通信。安装 **OpenShift Container Platform** 时，默认的 **OpenShift Kube API** 服务器证书由内部 **Red Hat OpenShift Container Platform** 集群证书颁发机构(CA)发布。如果需要，您可以添加或更改证书。

更改证书可能会影响受管集群和 **hub** 集群之间的通信。当您在安装产品前添加指定证书时，您可以避免使受管集群处于离线状态的问题。

以下列表包含一些您可能需要更新证书时的示例：

- 您要將负载均衡器的默认证书替换为您自己的证书。按照在 **OpenShift Container Platform** 文档中的 *添加 API 服务器证书* 的指导信息，添加指定证书为主机名 **api.<cluster_name>.<base_domain >** 来替换外部负载均衡器的默认 **API** 服务器证书。替换证书可能会导致一些受管集群进入离线状态。如果在升级证书后集群处于离线状态，请按照 *证书更改后离线对导入集群进行故障排除* 的说明进行操作。

注：在安装产品前添加指定证书有助于避免集群进入离线状态。

- 外部负载均衡器的证书即将过期，您需要替换它。完成以下步骤以替换证书：

1. 找到 **APIServer** 自定义资源，如下例所示：

```
apiVersion: config.openshift.io/v1
kind: APIServer
metadata:
  name: cluster
spec:
  audit:
    profile: Default
  servingCerts:
    namedCertificates:
      - names:
        - api.mycluster.example.com
      servingCertificate:
        name: old-cert-secret
```

2. 运行以下命令，在 **openshift-config** 命名空间中创建一个新 **secret**，其中包含现有证书和新证书的内容：

- a. 将旧证书复制到新证书中：

-

```
cp old.crt combined.crt
```

- b. 将新证书的内容添加到旧证书的副本中：

```
cat new.crt >> combined.crt
```

- c. 应用组合证书以创建 **secret**：

```
oc create secret tls combined-certs-secret --cert=combined.crt --key=old.key -n openshift-config
```

3. 更新您的 **APIServer** 资源，将组合证书引用为 **serviceCertificate**。

```
apiVersion: config.openshift.io/v1
kind: APIServer
metadata:
  name: cluster
spec:
  audit:
    profile: Default
  servingCerts:
    namedCertificates:
      - names:
        - api.mycluster.example.com
      servingCertificate:
        name: combined-cert-secret
```

4. 大约 **15** 分钟后，包含新证书和旧证书的 **CA** 捆绑包将传播到受管集群。

5. 输入以下命令在 **openshift-config** 命名空间中创建一个名为 **new-cert-secret** 的 **secret**，该 **secret** 只包括新证书信息：

```
oc create secret tls new-cert-secret --cert=new.crt --key=new.key -n openshift-config {code}
```

6. 通过更改 **serviceCertificate** 的名称以引用 **new-cert-secret** 来更新 **APIServer** 资源。您的资源可能类似以下示例：

```
apiVersion: config.openshift.io/v1
kind: APIServer
metadata:
  name: cluster
```

```
spec:
  audit:
    profile: Default
  servingCerts:
    namedCertificates:
      - names:
        - api.mycluster.example.com
      servingCertificate:
        name: new-cert-secret
```

大约 15 分钟后，旧证书会从 CA 捆绑包中删除，更改会自动传播到受管集群。

注：受管集群必须使用主机名 `api.<cluster_name>.<base_domain>` 来访问 hub 集群。您不能使用配置了其他主机名的命名证书。

1.5.15.2. 其他资源

- [添加 API 服务器证书](#)
- [证书更改后导入的集群离线故障排除](#)

1.5.16. 从管理中移除集群

当您从管理中删除使用 **multicluster engine operator** 创建的 **OpenShift Container Platform** 集群时，您可以分离或销毁它。分离集群会将其从管理中移除，但不会完全删除。如果要管理它，您可以再次导入它。只有集群处于 **Ready** 状态时方可使用这个选项。

以下流程在以下情况下从管理中删除集群：

- 您已删除集群，并希望从 **Red Hat Advanced Cluster Management** 中删除已删除的集群。
- 您要从管理中删除集群，但还没有删除集群。

重要：

- 销毁集群会将其从管理中移除，并删除集群的组件。

- 当您分离受管集群时，相关命名空间会被自动删除。不要将自定义资源放在这个命名空间中。
 - [使用控制台删除集群](#)
 - [使用命令行删除集群](#)
 - [删除集群后删除剩余的资源](#)
 - [删除集群后对 etcd 数据库进行碎片整理](#)

1.5.16.1. 使用控制台删除集群

在导航菜单中导航到 **Infrastructure > Clusters**，从您要从管理中删除的集群旁的选项菜单中选择 **Destroy cluster** 或 **Detach cluster**。

提示：您可以通过选择要分离或销毁的集群的复选框来分离或销毁多个集群，然后选择 **Detach** 或 **Destroy**。

注：如果您在管理的 **hub** 集群（称为 **local-cluster**）时尝试分离 **hub** 集群，请检查 **disableHubSelfManagement** 的默认设置是否为 **false**。此设置会导致 **hub** 集群在分离时重新导入自己并管理自己，并协调 **MultiClusterHub** 控制器。**hub** 集群可能需要几小时时间来完成分离过程并重新导入。

要在不需要等待进程完成的情况下重新导入 **hub** 集群，您可以输入以下命令来重启 **multiclusterhub-operator pod** 并更快地重新导入：

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

您可以通过将 **disableHubSelfManagement** 值改为 **true** 来更改 **hub** 集群的值，如 [在线安装](#) 中所述。

1.5.16.2. 使用命令行删除集群

要使用 **hub** 集群的命令行分离受管集群，请运行以下命令：

```
oc delete managedcluster $CLUSTER_NAME
```

要在分离后销毁受管集群，请运行以下命令：

```
oc delete clusterdeployment <CLUSTER_NAME> -n $CLUSTER_NAME
```

备注：

- 要防止销毁受管集群，在 **ClusterDeployment** 自定义资源中将 **spec.preserveOnDelete** 参数设置为 **true**。
- **disableHubSelfManagement** 的默认设置是 **false**。**false** 设置会导致 **hub** 集群（也称为 **'local-cluster'**）在分离时重新导入和管理自己，并协调 **MultiClusterHub** 控制器。

分离和重新导入过程可能需要几小时时间才能完成 **hub** 集群。如果要在等待进程完成后重新导入 **hub** 集群，您可以输入以下命令来重启 **multiclusterhub-operator pod** 并更快地重新导入：

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

您可以通过将 **disableHubSelfManagement** 值改为 **true** 来更改 **hub** 集群的值，使其不会自动导入。请参阅[在线安装](#)。

1.5.16.3. 删除集群后删除剩余的资源

如果受管集群上有剩余的资源，则需要额外的步骤以确保删除所有剩余的组件。需要这些额外步骤的情况，包括以下示例：

- 受管集群在完全创建前会被分离，但 **klusterlet** 等一些组件会保留在受管集群中。
- 在分离受管集群前，管理集群的 **hub** 丢失或销毁，因此无法从 **hub** 中分离受管集群。

- 当受管集群被分离后，受管集群将处于非在线状态。

如果其中一个情况适用于您试图分离的受管集群，则有些资源将无法从受管集群中删除。完成以下步骤以分离受管集群：

1. 确保配置了 **oc** 命令行界面。
2. 确保您在受管集群中配置了 **KUBECONFIG**。

如果运行 **oc get ns | grep open-cluster-management-agent**，您应该看到两个命名空间：

```
open-cluster-management-agent      Active 10m
open-cluster-management-agent-addon Active 10m
```

3. 运行以下命令以删除剩余的资源：

```
oc delete namespaces open-cluster-management-agent open-cluster-management-agent-addon --wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc delete crds --wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc patch crds --type=merge -p '{"metadata":{"finalizers": []}]'
```

4. 运行以下命令，以确保命名空间和所有打开的集群管理 **crds** 均已被删除：

```
oc get crds | grep open-cluster-management.io | awk '{print $1}'
oc get ns | grep open-cluster-management-agent
```

1.5.16.4. 删除集群后对 **etcd** 数据库进行碎片整理

拥有多个受管集群可能会影响 **hub** 集群中 **etcd** 数据库的大小。在 **OpenShift Container Platform 4.8** 中，当删除受管集群时，**hub** 集群中的 **etcd** 数据库不会被自动减小。在某些情况下，**etcd** 数据库可能会耗尽空间。此时会显示一个错误 **etcdserver: mvcc: database space exceeded**。要更正此错误，请通过压缩数据库历史记录并对 **etcd** 数据库进行碎片整理来减小 **etcd** 数据库的大小。

注：对于 **OpenShift Container Platform** 版本 **4.9** 及更新的版本，**etcd Operator** 会自动清理磁盘并压缩 **etcd** 历史记录。不需要人工干预。以下流程适用于 **OpenShift Container Platform** 版本 **4.8** 及更早版本。

通过完成以下步骤，压缩 **etcd** 历史记录并整理 **hub** 集群中 **etcd** 数据库的碎片。

1.5.16.4.1. 先决条件

- 安装 **OpenShift CLI (oc)** 。
- 以具有 **cluster-admin** 特权的用户身份登录。

1.5.16.4.2. 流程

1. 压缩 **etcd** 历史记录。
 - a. 打开到 **etcd** 成员的远程 **shell** 会话，例如：

```
$ oc rsh -n openshift-etcd etcd-control-plane-0.example.com etcdctl endpoint status --cluster -w table
```

- b. 运行以下命令来压缩 **etcd** 历史记录：

```
sh-4.4#etcdctl compact $(etcdctl endpoint status --write-out="json" | egrep -o '"revision":[0-9]*' | egrep -o '[0-9]*' -m1)
```

输出示例

```
$ compacted revision 158774421
```

2. 清理 **etcd** 数据库并清除任何 **NOSPACE** 警报，如[分离 etcd 数据](#) 中所述。

1.6. 发现服务简介

您可以发现 **OpenShift Cluster Manager** 可用的 **OpenShift 4** 集群。发现后，您可以导入集群进行管理。发现服务使用 **Discover Operator** 进行后端和控制台使用。

您必须具有 **OpenShift Cluster Manager** 凭据。如果需要 [创建凭证](#)，请参阅为 **Red Hat OpenShift Cluster Manager** 创建凭证。

需要的访问权限：**Administrator**

- [使用控制台配置发现](#)
- [使用 CLI 配置发现](#)

1.6.1. 使用控制台配置发现

使用产品控制台启用发现。

需要的访问权限：访问创建凭证的命名空间。

1.6.1.1. 先决条件

- 您需要一个凭证。请参阅为 **Red Hat OpenShift Cluster Manager** [创建凭证](#) 以连接到 **OpenShift Cluster Manager**。

1.6.1.2. 配置发现

在控制台中配置 **Discovery** 以查找集群。您可以使用单独的凭证创建多个 **DiscoveryConfig** 资源。按照控制台中的说明操作。

1.6.1.3. 查看发现的集群

在设置凭证并发现集群以导入后，您可以在控制台中查看它们。

1. 点 **Clusters > Discovered cluster**

2.

使用以下信息查看填充的表：

- **Name** 是 OpenShift Cluster Manager 中指定的显示名称。如果集群没有显示名称，则会显示基于集群控制台 URL 生成的名称。如果 OpenShift Cluster Manager 缺少控制台 URL，或手动修改了控制台 URL，则会显示集群外部 ID。
- **Namespace** 是您创建凭证和发现集群的命名空间。
- **Type** 是发现的集群 Red Hat OpenShift 类型。
- **Distribution version** 是发现的集群 Red Hat OpenShift 版本。
- **基础架构供应商**是已发现集群的云供应商。
- **最后活跃**是发现的集群最后一次活跃的时间。
- 当发现的集群被创建时为 **Created**。
- 当发现的集群被发现时为 **Discovered**。

3.

您还可以搜索表中的任何信息。例如，要只显示特定命名空间中的发现集群，请搜索该命名空间。

4.

现在，您可以点 **Import cluster** 创建受管集群。请参阅[导入发现的集群](#)。

1.6.1.4. 导入发现的集群

发现集群后，您可以导入控制台的 **Discovered clusters** 选项卡中出现的集群。

1.6.1.5. 先决条件

您需要访问用于配置 **Discovery** 的命名空间。

1.6.1.6. 导入发现的集群

1. 进入到现有 **Clusters** 页面并点 **Discovered clusters** 选项卡。
2. 在发现的集群表中找到您要导入的集群。
3. 在选项菜单中选择 **Import cluster**。
4. 对于发现的集群，您可以使用文档手动导入，或者您可以自动选择导入集群。
5. 要使用凭证或 **Kubeconfig** 文件自动导入，请复制并粘贴内容。
6. 点 **Import**。

1.6.2. 使用 CLI 启用发现

使用 **CLI** 启用发现以查找 **Red Hat OpenShift Cluster Manager** 可用的集群。

需要的访问权限：**Administrator**

1.6.2.1. 先决条件

- 创建用于连接到 **Red Hat OpenShift Cluster Manager** 的凭证。

1.6.2.2. 发现设置和进程

注：**DiscoveryConfig** 必须命名为 **discovery**，且必须与所选凭证在同一命名空间中创建。请参见以下 **DiscoveryConfig** 示例：

```
apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveryConfig
```

```

metadata:
  name: discovery
  namespace: <NAMESPACE_NAME>
spec:
  credential: <SECRET_NAME>
  filters:
    lastActive: 7
    openshiftVersions:
      - "4.10"
      - "4.11"
      - "4.8"

```

1. 将 **SECRET_NAME** 替换为之前设置的凭证。
2. 将 **NAMESPACE_NAME** 替换为 **SECRET_NAME** 的命名空间。
3. 输入集群最后一次活动（以天为单位）进行发现的最大时间。例如，带有 **lastActive: 7** 的集群，最后 7 天内活跃的集群会被发现。
4. 输入要作为字符串列表发现的 **Red Hat OpenShift** 集群的版本。注：**openshiftVersions** 列表中的每个条目都指定了一个 **OpenShift** 主版本和次版本。例如，指定 **"4.11"** 将包括 **OpenShift** 版本 4.11 的所有补丁版本，如 **4.11.1**、**4.11.2**。

1.6.2.3. 查看发现的集群

通过运行 `oc get discoveredclusters -n <namespace>`（其中 **namespace** 是发现凭证存在的命名空间）来查看发现的集群。

1.6.2.3.1. DiscoveredClusters

对象由 **Discovery** 控制器创建。这些 **DiscoveredClusters** 使用在 **DiscoveryConfig** `discoveredclusters.discovery.open-cluster-management.io` API 中指定的过滤器和凭证来代表 **OpenShift Cluster Manager** 中找到的集群。**name** 的值是集群外部 ID：

```

apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveredCluster
metadata:
  name: fd51aafa-95a8-41f7-a992-6fb95eed3c8e
  namespace: <NAMESPACE_NAME>
spec:
  activity_timestamp: "2021-04-19T21:06:14Z"
  cloudProvider: vsphere
  console: https://console-openshift-console.apps.qe1-vmware-pkt.dev02.red-chesterfield.com

```

```

creation_timestamp: "2021-04-19T16:29:53Z"
credential:
  apiVersion: v1
  kind: Secret
  name: <SECRET_NAME>
  namespace: <NAMESPACE_NAME>
display_name: qe1-vmware-pkt.dev02.red-chesterfield.com
name: fd51aafa-95a8-41f7-a992-6fb95eed3c8e
openshiftVersion: 4.10
status: Stale

```

1.7. 托管 CONTROL PLANE（技术预览）

使用 **multicluster engine operator** 集群管理，您可以使用两个不同的 **control plane** 配置部署 **OpenShift Container Platform** 集群：独立或托管的 **control plane**。独立配置使用专用虚拟机或物理机器来托管 **OpenShift Container Platform control plane**。使用 **OpenShift Container Platform** 托管 **control plane**，您可以在托管集群中创建 **pod** 作为 **control plane**，而无需为每个 **control plane** 专用物理机器。

为 **OpenShift Container Platform** 托管 **control plane** 作为 **Amazon Web Services (AWS)** 和裸机上的技术预览功能提供。您可以在 **OpenShift Container Platform** 版本 **4.10.7** 及更高版本中托管 **control plane**。

注：在托管 **control plane** 的同一平台上运行 **hub** 集群和 **worker**。

control plane 作为单一命名空间中包含的 **pod** 运行，并与托管的 **control plane** 集群关联。当 **OpenShift Container Platform** 创建这种类型的托管集群时，它会创建一个独立于 **control plane** 的 **worker** 节点。

托管 **control plane** 集群有几个优点：

- 通过删除对专用 **control plane** 节点的需求来降低成本
- 引入 **control plane** 和工作负载的隔离，改进了隔离并减少可能需要更改的配置错误
- 通过删除 **control plane** 节点 **bootstrap** 的要求来减少集群创建时间
- 支持 **turn-key** 部署或完全自定义的 **OpenShift Container Platform** 置备

有关托管 **control plane** 的更多信息，请继续阅读以下主题：

- [在 AWS 上配置托管集群（技术预览）](#)
- [在 AWS 上管理托管的 **control plane**（技术预览）](#)
- [在裸机上配置托管集群（技术预览）](#)
- [在裸机上管理托管的 **control plane** 集群（技术预览）](#)
- [禁用托管的控制功能（技术预览）](#)

1.7.1. 在 AWS 上配置托管集群（技术预览）

要配置托管的 **control plane**，您需要托管集群和一个托管的集群。通过使用 **hypershift-addon** 受管集群附加组件在现有受管集群上部署 **HyperShift operator**，您可以将该集群启用为托管集群，并开始创建托管集群。

multicluster engine operator 2.2 仅支持默认的 **local-cluster** 和 **hub** 集群作为托管集群。

托管 **control plane** 是一个技术预览功能，因此相关组件默认是禁用的。

您可以通过将现有集群配置为充当托管集群来部署托管 **control plane**。托管集群是托管 **control plane** 的 **Red Hat OpenShift Container Platform** 集群。**Red Hat Advanced Cluster Management 2.7** 可以使用 **hub** 集群（也称为 **local-cluster**）作为托管集群。请参阅以下主题以了解如何将 **local-cluster** 配置为托管集群。

最佳实践：确保在托管 **control plane** 的同一平台上运行 **hub** 集群和 **worker**。

- [先决条件](#)

- [创建 Amazon Web Services S3 存储桶和 S3 OIDC secret](#)
- [创建可路由的公共区](#)
- [启用外部 DNS](#)
- [启用 AWS PrivateLink](#)
- [启用托管的 control plane 功能](#)
- [为 local-cluster 手动启用 hypershift-addon 受管集群附加组件](#)
- [安装托管的 control plane CLI](#)
- [其他资源](#)

1.7.1.1. 先决条件

您必须具有以下先决条件才能配置托管集群：

- **Kubernetes operator 2.2** 及更新版本的多集群引擎安装在 **OpenShift Container Platform** 集群中。安装 **Red Hat Advanced Cluster Management** 时会自动安装 **multicluster engine operator**。在没有 **Red Hat Advanced Cluster Management** 作为来自 **OpenShift Container Platform OperatorHub** 的 **Operator** 的情况下，也可以安装 **multicluster engine operator**。
- **multicluster engine operator** 必须至少有一个受管 **OpenShift Container Platform** 集群。**local-cluster** 在多集群引擎 **operator 2.2** 及更新的版本中自动导入。有关 **local-cluster** 的更多信息，请参阅[高级配置](#)。您可以运行以下命令来检查 **hub** 集群的状态：

```
oc get managedclusters local-cluster
```

- 托管集群至少有 **3** 个 **worker** 节点来运行 **HyperShift Operator**。

- **AWS 命令行界面。**

1.7.1.2. 创建 Amazon Web Services S3 存储桶和 S3 OIDC secret

如果您计划在 **AWS** 上创建和管理托管集群，请完成以下步骤：

- 1.

创建一个 **S3** 存储桶，其具有集群托管 **OIDC** 发现文档的公共访问权限。

- 要在 **us-east-1** 区域中创建存储桶，请输入以下代码：

```

BUCKET_NAME=<your_bucket_name>
aws s3api create-bucket --bucket $BUCKET_NAME
aws s3api delete-public-access-block --bucket $BUCKET_NAME
echo '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::${BUCKET_NAME}/*"
    }
  ]
}' | envsubst > policy.json
aws s3api put-bucket-policy --bucket $BUCKET_NAME --policy file://policy.json

```

- 要在 **us-east-1** 区域以外的区域中创建存储桶，请输入以下代码：

```

BUCKET_NAME=your-bucket-name
REGION=us-east-2
aws s3api create-bucket --bucket $BUCKET_NAME \
  --create-bucket-configuration LocationConstraint=$REGION \
  --region $REGION
aws s3api delete-public-access-block --bucket $BUCKET_NAME
echo '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::${BUCKET_NAME}/*"
    }
  ]
}'

```

```

    ]
  } | envsubst > policy.json
aws s3api put-bucket-policy --bucket $BUCKET_NAME --policy file://policy.json

```

2. 为 **HyperShift Operator** 创建名为 **hypershift-operator-oidc-provider-s3-credentials** 的 **OIDC S3 secret**。
3. 将 **secret** 保存到 **local-cluster** 命名空间中。
4. 请参阅下表来验证 **secret** 是否包含以下字段：

字段名称	描述
bucket	包含对 HyperShift 集群托管 OIDC 发现文档的公共访问权限的 S3 存储桶。
credentials	对包含可以访问存储桶的 default 配置集凭证的文件的引用。默认情况下，HyperShift 仅使用 default 配置集来运行 bucket 。
region	指定 S3 存储桶的区域。

以下示例显示了 **AWS secret** 模板示例：

```

oc create secret generic hypershift-operator-oidc-provider-s3-credentials --from-
file=credentials=$HOME/.aws/credentials --from-literal=bucket=<s3-bucket-for-hypershift> --
from-literal=region=<region> -n local-cluster

```

注：不会自动启用 **secret** 的恢复备份。运行以下命令添加启用 **hypershift-operator-oidc-provider-s3-credentials secret** 的标签，以便为灾难恢复进行备份：

```

oc label secret hypershift-operator-oidc-provider-s3-credentials -n local-cluster cluster.open-
cluster-management.io/backup=true

```

1.7.1.3. 创建可路由的公共区

要访问客户机集群中的应用程序，必须路由公共区。如果 **public** 区域存在，请跳过这一步。否则，公共区将影响现有功能。

运行以下命令为集群 **DNS** 记录创建一个公共区：

```
BASE_DOMAIN=www.example.com
aws route53 create-hosted-zone --name $BASE_DOMAIN --caller-reference $(whoami)-$(date --rfc-3339=date)
```

1.7.1.4. 启用外部 DNS

如果您计划使用服务级别 **DNS**（外部 **DNS**）置备托管的 **control plane** 集群，请完成以下步骤：

1. 为 **HyperShift Operator** 创建一个 **AWS** 凭证 **secret**，并在 **local-cluster** 命名空间中将其命名为 **hypershift-operator-external-dns-credentials**。

2. 请参阅下表来验证 **secret** 是否包含所需字段：

字段名称	描述	可选或必需的
provider	管理服务级别 DNS 区的 DNS 提供程序。	必需
domain-filter	服务级别域。	必需
credentials	支持所有外部 DNS 类型的凭据文件。	使用 AWS 密钥时的可选
aws-access-key-id	凭证访问密钥 id。	使用 AWS DNS 服务时的可选
aws-secret-access-key	凭证访问密钥 secret。	使用 AWS DNS 服务时的可选

如需更多信息，请参阅 **HyperShift** 文档中的 **外部 DNS**。以下示例显示了 **hypershift-operator-external-dns-credentials secret** 模板示例：

```
oc create secret generic hypershift-operator-external-dns-credentials --from-literal=provider=aws --from-literal=domain-filter=service.my.domain.com --from-file=credentials=<credentials-file> -n local-cluster
```

注：不会自动启用 **secret** 的恢复备份。运行以下命令添加启用 **hypershift-operator-external-dns-credentials secret** 的标签，以便备份灾难恢复：

```
oc label secret hypershift-operator-external-dns-credentials -n local-cluster cluster.open-cluster-management.io/backup=""
```

1.7.1.5. 启用 AWS PrivateLink

如果您计划使用 **PrivateLink** 在 **AWS** 平台上置备托管的 **control plane** 集群，请完成以下步骤：

1. 为 **HyperShift Operator** 创建一个 **AWS** 凭证 **secret**，并将其命名为 **hypershift-operator-private-link-credentials**。**secret** 必须驻留在受管集群命名空间中，该命名空间用作托管集群。如果使用 **local-cluster**，请在 **local-cluster** 命名空间中创建 **secret**。
2. 请参阅下表确认 **secret** 包含必填字段：

字段名称	描述	可选或必需的
region	用于私有链接的区域	必填
aws-access-key-id	凭证访问密钥 id。	必填
aws-secret-access-key	凭证访问密钥 secret。	必填

如需更多信息，请参阅 **HyperShift** 文档中的 **部署 AWS 私有集群**。以下示例显示了 **hypershift-operator-private-link-credentials secret** 模板示例：

```
oc create secret generic hypershift-operator-private-link-credentials --from-literal=aws-access-key-id=<aws-access-key-id> --from-literal=aws-secret-access-key=<aws-secret-access-key> --from-literal=region=<region> -n local-cluster
```

注：不会自动启用 **secret** 的恢复备份。运行以下命令添加启用 **hypershift-operator-private-link-credentials secret** 的标签，以便备份灾难恢复：

```
oc label secret hypershift-operator-private-link-credentials -n local-cluster cluster.open-cluster-management.io/backup=""
```

1.7.1.6. 启用托管的 control plane 功能

托管的 **control plane** 功能默认禁用。启用该功能还会自动启用 **hypershift-addon** 受管集群附加组件。您可以运行以下命令来启用该功能：

```
oc patch mce multiclusterengine --type=merge -p '{"spec":{"overrides":{"components":[{"name":"hypershift-preview","enabled": true}]}}}'
```

运行以下命令，以验证 **MultiClusterEngine** 自定义资源中是否启用了 **hypershift-preview** 和 **hypershift-local-hosting** 功能。

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

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

1.7.1.6.1. 为 local-cluster 手动启用 hypershift-addon 受管集群附加组件

启用托管的 **control plane** 功能会自动启用 **hypershift-addon** 受管集群附加组件。如果您需要手动启用 **hypershift-addon** 受管集群附加组件，请完成以下步骤，使用 **hypershift-addon** 在 **local-cluster** 上安装 **HyperShift Operator**：

1. 通过创建一个类似以下示例的文件来创建 **ManagedClusterAddon HyperShift** 附加组件：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: hypershift-addon
  namespace: local-cluster
spec:
  installNamespace: open-cluster-management-agent-addon
```

2. 运行以下命令来应用该文件：

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

使用您创建的文件名称替换 **filename**。

3.

运行以下命令确认已安装 **hypershift-addon**：

```
oc get managedclusteraddons -n local-cluster hypershift-addon
```

4.

如果安装了附加组件，输出类似以下示例：

```
NAME          AVAILABLE DEGRADED PROGRESSING
hypershift-addon True
```

您的 **HyperShift** 附加组件已安装，托管集群可用于创建和管理 **HyperShift** 集群。

1.7.1.7. 安装托管的 control plane CLI

托管的 **control plane (HyperShift) CLI** 用于创建和管理 **OpenShift Container Platform** 托管 **control plane** 集群。启用托管的 **control plane** 功能后，您可以通过完成以下步骤来安装托管的 **control plane CLI**：

1.

在 **OpenShift Container Platform** 控制台中点 **Help** 图标 **>** **Command Line Tools**。

2.

为您的平台点 **Download hypershift CLI**。

注：下载只有在启用了 **hypershift-preview** 功能时才可见。

3.

运行以下命令来解包下载的归档：

```
tar xvzf hypershift.tar.gz
```

4.

运行以下命令使二进制文件可执行：

```
chmod +x hypershift
```

5.

运行以下命令，将二进制文件移到路径的目录中：

```
sudo mv hypershift /usr/local/bin/.
```

现在，您可以使用 `hypershift create cluster aws --help` 命令来创建和管理托管集群。使用以下命令列出可用的参数：

```
hypershift create cluster aws --help
```

1.7.1.8. 其他资源

- 如需有关 **AWS** 凭证 **secret** 的更多信息，请参阅 **HyperShift** 文档中的 [部署 AWS 私有集群](#)。
- 有关外部 **DNS** 的更多信息，请参阅 **HyperShift** 文档中的 [外部 DNS](#)。
- 现在，您可以部署 **SR-IOV Operator**。请参阅为托管 **control plane** 部署 **SR-IOV Operator**，以了解更多有关部署 **SR-IOV Operator** 的信息。

1.7.2. 在 **AWS** 上管理托管的 **control plane** 集群（技术预览）

您可以使用 **multicluster engine for Kubernetes operator** 控制台创建 **Red Hat OpenShift Container Platform** 托管的集群。托管 **control plane** 在 **Amazon Web Services (AWS)** 上作为技术预览提供。如果在 **AWS** 上使用托管的 **control plane**，您可以使用控制台创建托管集群，也可以使用控制台或 **CLI** 导入托管集群。

- [先决条件](#)
- [使用控制台在 **AWS** 上创建托管集群](#)
- [在 **AWS** 上导入托管的 **control plane** 集群](#)
- [访问 **AWS** 上的托管集群](#)
- [销毁 **AWS** 上的托管集群](#)

1.7.2.1. 先决条件

您必须先配置托管的 **control plane**，然后才能创建托管的 **control plane** 集群。如需更多信息，请参
[阅配置托管的 control plane](#)（技术预览）。

1.7.2.2. 使用控制台在 AWS 上创建托管的 control plane 集群

要从多集群引擎 **operator** 控制台创建托管的 **control plane** 集群，请进入到 **Infrastructure > Clusters**。在 **Clusters** 页面中，点 **Create cluster > Amazon Web Services > Hosted** 并完成控制台中的步骤。

注：在到达控制台后，使用提供的基本信息来在命令行中创建集群。

1.7.2.3. 使用命令行在 AWS 上部署托管集群

设置 **HyperShift** 命令行界面并将 **local-cluster** 启用为托管集群后，您可以通过完成以下步骤在 **AWS** 上部署托管集群：

1. 按如下方式设置环境变量，根据需要将变量替换为您的凭证：

```
export REGION=us-east-1
export CLUSTER_NAME=clc-name-hs1
export INFRA_ID=clc-name-hs1
export BASE_DOMAIN=dev09.red-chesterfield.com
export AWS_CREDS=$HOME/name-aws
export PULL_SECRET=/Users/username/pull-secret.txt
export BUCKET_NAME=acmqe-hypershift
export BUCKET_REGION=us-east-1
```

2. 验证 **CLUSTER_NAME** 和 **INFRA_ID** 是否有相同的值，否则集群可能无法在 **Kubernetes operator** 控制台的多集群引擎中显示。要查看每个变量的描述，请运行以下命令

```
hypershift create cluster aws --help
```

3. 验证您是否已登录到 **hub** 集群。

4. 运行以下命令来创建托管集群：

```
hypershift create cluster aws \
  --name $CLUSTER_NAME \
  --infra-id $INFRA_ID \
  --aws-creds $AWS_CREDS \
```

```
--pull-secret $PULL_SECRET \
--region $REGION \
--generate-ssh \
--node-pool-replicas 3 \
--namespace <hypershift-hosting-service-cluster>
```

注：默认情况下，所有 **HostedCluster** 和 **NodePool** 自定义资源都是在 **集群命名空间** 中创建的。如果指定了 `--namespace <namespace>` 参数，则会在您选择的命名空间中创建 **HostedCluster** 和 **NodePool** 自定义资源。

5.

您可以运行以下命令来检查托管集群的状态：

```
oc get hostedclusters -n <hypershift-hosting-service-cluster>
```

1.7.2.4. 在 AWS 上导入托管的 control plane 集群

您可以使用控制台导入托管的 **control plane** 集群。

1.

进入 **Infrastructure > Clusters** 并选择您要导入的托管集群。

2.

点 **Import hosted cluster**。

注：对于 *发现的* 托管集群，您还可以从控制台导入，但集群必须处于可升级状态。如果托管集群没有处于可升级状态，则在集群中导入会被禁用，因为托管的 **control plane** 不可用。点 **Import** 开始进程。当集群接收更新时，状态为 **Importing**，然后变为 **Ready**。

您还可以通过完成以下步骤，使用 **CLI** 在 **AWS** 上导入托管的 **control plane** 集群：

1.

运行以下命令，在 **HostedCluster** 自定义资源中添加注解：

```
oc edit hostedcluster <cluster_name> -n clusters
```

将 `<cluster_name>` 替换为托管集群的名称。

2.

运行以下命令，将注解添加到 **HostedCluster** 自定义资源中：

```
cluster.open-cluster-management.io/hypershiftdeployment: local-cluster/<cluster_name>
cluster.open-cluster-management.io/managedcluster-name: <cluster_name>
```

将 **<cluster_name >** 替换为托管集群的名称。

3.

使用以下示例 **YAML** 文件创建 **ManagedCluster** 资源：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    import.open-cluster-management.io/hosting-cluster-name: local-cluster
    import.open-cluster-management.io/klusterlet-deploy-mode: Hosted
    open-cluster-management/created-via: other
  labels:
    cloud: auto-detect
    cluster.open-cluster-management.io/clusterset: default
    name: <cluster_name>
    vendor: OpenShift
    name: <cluster_name>
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

将 **<cluster_name >** 替换为托管集群的名称。

4.

运行以下命令以应用资源：

```
oc apply -f <file_name>
```

将 **<file_name>** 替换为您在上一步中创建的 **YAML** 文件名。

5.

使用以下示例 **YAML** 文件创建 **KlusterletAddonConfig** 资源。这只适用于 **Red Hat Advanced Cluster Management**。如果您只安装了 **multicluster engine operator**，请跳过这一步：

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster_name>
  namespace: <cluster_name>
spec:
  clusterName: <cluster_name>
```

```

clusterNamespace: <cluster_name>
clusterLabels:
  cloud: auto-detect
  vendor: auto-detect
applicationManager:
  enabled: true
certPolicyController:
  enabled: true
iamPolicyController:
  enabled: true
policyController:
  enabled: true
searchCollector:
  enabled: false

```

将 `<cluster_name>` 替换为托管集群的名称。

6.

运行以下命令以应用资源：

```
oc apply -f <file_name>
```

将 `<file_name>` 替换为您在上一步中创建的 **YAML** 文件名。

7.

导入过程完成后，您的托管集群会在控制台中可见。您还可以运行以下命令来检查托管集群的状态：

```
oc get managedcluster <cluster_name>
```

1.7.2.5. 访问 AWS 上的托管集群

托管 **control plane** 集群的访问 **secret** 存储在 **hypershift-management-cluster** 命名空间中。了解以下 **secret** 名称格式：

- **kubeconfig secret: & It;hostingNamespace>-<name>-admin-kubeconfig (clusters-hypershift-demo-admin-kubeconfig)**
- **kubeadmin password secret: & It;hostingNamespace>-<name>-kubeadmin-password (clusters-hypershift-demo-kubeadmin-password)**

1.7.2.6. 销毁 AWS 上的托管集群

要销毁托管的集群及其受管集群资源，请完成以下步骤：

1. 运行以下命令来删除托管集群及其后端资源：

```
hypershift destroy cluster aws --name <cluster_name> --infra-id <infra_id> --aws-creds  
<aws-credentials> --base-domain <base_domain> --destroy-cloud-resources
```

根据需要替换名称。

2. 运行以下命令，删除 **multicluster engine operator** 上的受管集群资源：

```
oc delete managedcluster <cluster_name>
```

将 **cluster_name** 替换为集群的名称。

1.7.3. 在裸机上配置托管集群（技术预览）

要配置托管的 **control plane**，您需要一个 *hosting*（进行托管）的集群和一个 *hosted*（被托管）的集群。您可以使用 **hypershift** 附加组件将受管集群启用为托管集群，以将 **HyperShift Operator** 部署到该集群中。然后，您可以开始创建托管集群。

multicluster engine operator 2.2 仅支持默认的 **local-cluster** 和 **hub** 集群作为托管集群。

托管 **control plane** 是一个技术预览功能，因此相关组件默认是禁用的。

您可以通过将集群配置为充当托管集群来部署托管 **control plane**。托管的集群是托管 **control plane** 的 **OpenShift Container Platform** 集群。

在 **Red Hat Advanced Cluster Management 2.7** 中，您可以使用受管 **hub** 集群（也称为 **local-cluster**），作为托管集群。

重要：

- 在托管 **control plane** 的同一平台上运行 **hub** 集群和 **worker**。
- 要在裸机上置备托管的 **control plane**，您可以使用 **Agent** 平台。**Agent** 平台使用中央基础架构管理服务将 **worker** 节点添加到托管集群。
- 每个裸机主机都必须使用中央基础架构管理提供的发现镜像启动。您可以使用 **Cluster-Baremetal-Operator** 手动启动主机或通过自动化启动主机。每个主机启动后，它运行一个 **Agent** 进程来发现主机详情并完成安装。**Agent** 自定义资源代表每个主机。
- 当使用 **Agent** 平台创建托管集群时，**HyperShift** 在托管的 **control plane** 命名空间中安装 **Agent Cluster API** 供应商。
- 当您扩展节点池时，会创建一个机器。**Cluster API** 供应商找到一个经过批准的代理，正在传递验证，当前没有被使用，并满足节点池规格中指定的要求。您可以通过检查其状态和条件来监控代理的安装。
- 当您缩减节点池时，代理会从对应的集群中绑定。在重复使用集群前，您必须使用 **Discovery** 镜像更新节点数量来重启它们。
 - [先决条件](#)
 - [配置 DNS](#)
 - [其他资源](#)

1.7.3.1. 先决条件

您必须具有以下先决条件才能配置托管集群：

- 您需要 **multicluster engine for Kubernetes operator 2.2** 及之后的版本安装在 **OpenShift Container Platform** 集群中。安装 **Red Hat Advanced Cluster Management** 时会自动安装 **multicluster engine operator**。您还可以在没有 **Red Hat Advanced Cluster Management** 作为 **OpenShift Container Platform OperatorHub** 的 **Operator** 的情况下安装 **multicluster engine operator**。

- 您需要 **multicluster engine operator**，必须至少有一个受管 **OpenShift Container Platform** 集群。**local-cluster** 在多集群引擎 **operator 2.2** 及更新的版本中自动导入。有关 **local-cluster** 的更多信息，请参阅[高级配置](#)。您可以运行以下命令来检查 **hub** 集群的状态：

```
oc get managedclusters local-cluster
```

- 您需要至少 **3** 个 **worker** 节点的托管集群来运行 **HyperShift Operator**。
- 您必须启用托管的 **control plane** 功能。如需更多信息，请参阅[启用托管的 control plane 功能](#)。

1.7.3.2. 配置 DNS

托管集群的 **API** 服务器作为 **NodePort** 服务公开。必须存在 `api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN}` 的 **DNS** 条目，指向可访问 **API** 服务器的目的地。

DNS 条目可以像一个记录一样简单，指向运行托管 **control plane** 的受管集群中的一个节点。该条目也可以指向部署的负载均衡器，以将传入的流量重定向到 **Ingress pod**。

请参见以下 **DNS** 配置示例：

```
api.example.krnl.es. IN A 192.168.122.20
api.example.krnl.es. IN A 192.168.122.21
api.example.krnl.es. IN A 192.168.122.22
api-int.example.krnl.es. IN A 192.168.122.20
api-int.example.krnl.es. IN A 192.168.122.21
api-int.example.krnl.es. IN A 192.168.122.22
`*`.apps.example.krnl.es. IN A 192.168.122.23
```

1.7.3.3. 其他资源

- 有关中央基础架构管理服务简介，请参阅 [Kube API - 入门指南](#)。
- 有关负载均衡器的更多信息，请参阅 [用户置备的基础架构的负载平衡要求](#)。
- 现在，您可以部署 **SR-IOV Operator**。请参阅[为托管 control plane 部署 SR-IOV Operator](#)，以了解更多有关部署 **SR-IOV Operator** 的信息。

1.7.4. 在裸机上管理托管的 **control plane** 集群（技术预览）

您可以使用 **Kubernetes operator** 的多集群引擎来创建和管理 **Red Hat OpenShift Container Platform** 托管的集群。托管 **control plane** 在 **Amazon Web Services (AWS)**和裸机上作为技术预览提供。

- [先决条件](#)
- [在裸机上创建托管集群](#)
- [创建 **InfraEnv**](#)
- [添加代理](#)
- [访问托管集群](#)
- [扩展 **NodePool** 对象](#)
- [处理入口](#)
- [为托管集群启用节点自动扩展](#)
- [验证托管集群创建](#)
- [在裸机上销毁托管集群](#)
- [其他资源](#)

1.7.4.1. 先决条件

您必须先为裸机配置托管的 **control plane**，然后才能创建托管的 **control plane** 集群。如需更多信息，请参阅[在裸机上配置托管集群（技术预览）](#)。

1.7.4.2. 使用控制台在裸机代理上创建托管的 **control plane** 集群

要从多集群引擎 **operator** 控制台创建托管的 **control plane** 集群，请进入到 **Infrastructure > Clusters**。在 **Clusters** 页面中，点 **Create cluster > Host Inventory > Hosted control plane**，并完成控制台中的步骤。

重要：当您创建集群时，多集群引擎 **operator** 控制器为集群及其资源创建一个命名空间。确保只在该命名空间中包含该集群实例的资源。销毁集群会删除命名空间和所有资源。

如果要将集群添加到现有的集群集中，则需要在集群设置上具有正确的权限来添加它。如果在创建集群时没有 **cluster-admin** 权限，则必须选择一个具有 **clusterset-admin** 权限的集群集。如果您在指定的集群集中没有正确的权限，集群创建会失败。如果您没有任何集群设置选项，请联络您的集群管理员，为您提供 **clusterset-admin** 权限。

每个受管集群都必须与受管集群集关联。如果您没有将受管集群分配给 **ManagedClusterSet**，则会自动添加到 **default** 受管集群集中。

发行镜像标识用于创建集群的 **OpenShift Container Platform** 镜像的版本。托管的 **control plane** 集群必须使用其中一个提供的发行镜像。

基础架构环境中提供的代理信息会自动添加到代理字段中。您可以使用现有信息，覆盖它，或者添加信息（如果要启用代理）。以下列表包含创建代理所需的信息：

- **HTTP 代理 URL**：用作 **HTTP** 流量的代理的 **URL**。
- **HTTPS 代理 URL**：用于 **HTTPS** 流量的安全代理 **URL**。如果没有提供值，则使用相同的值 **HTTP Proxy URL**，用于 **HTTP** 和 **HTTPS**。
- **无代理域**：应当绕过代理的以逗号分隔的域列表。使用一个句点 (.) 开始的域名，包含该域中的所有子域。添加一个星号 * 以绕过所有目的地的代理。



Additional trust bundle : 访问镜像 **registry** 所需的证书文件内容。

注：您必须运行随集群提供的集群详情的 **oc** 命令，以导入集群。创建集群时，它不会使用 **Red Hat Advanced Cluster Management** 管理自动配置。

1.7.4.3. 使用命令行在裸机上创建托管集群

验证您是否为集群配置了默认存储类。否则，可能会以待处理的 **PVC** 结束。

1.

输入以下命令：

```
export CLUSTERS_NAMESPACE="clusters"
export HOSTED_CLUSTER_NAME="example"
export HOSTED_CONTROL_PLANE_NAMESPACE="${CLUSTERS_NAMESPACE}-
${HOSTED_CLUSTER_NAME}"
export BASEDOMAIN="krnl.es"
export PULL_SECRET_FILE=$PWD/pull-secret
export MACHINE_CIDR=192.168.122.0/24
# Typically the namespace is created by the hypershift-operator
# but agent cluster creation generates a capi-provider role that
# needs the namespace to already exist
oc create ns ${HOSTED_CONTROL_PLANE_NAMESPACE}

hypershift create cluster agent \
  --name=${HOSTED_CLUSTER_NAME} \
  --pull-secret=${PULL_SECRET_FILE} \
  --agent-namespace=${HOSTED_CONTROL_PLANE_NAMESPACE} \
  --base-domain=${BASEDOMAIN} \
  --api-server-address=api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN} \
```

2.

片刻后，输入以下命令验证托管的 **control plane pod** 是否正在运行：

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get pods
```

输出示例

NAME	READY	STATUS	RESTARTS	AGE
capi-provider-7dcf5fc4c4-nr9sq	1/1	Running	0	4m32s
catalog-operator-6cd867cc7-phb2q	2/2	Running	0	2m50s
certified-operators-catalog-884c756c4-zdt64	1/1	Running	0	2m51s
cluster-api-f75d86f8c-56wfz	1/1	Running	0	4m32s
cluster-autoscaler-7977864686-2rz4c	1/1	Running	0	4m13s

```

cluster-network-operator-754cf4ffd6-lwfm2    1/1    Running 0    2m51s
cluster-policy-controller-784f995d5-7cbrz    1/1    Running 0    2m51s
cluster-version-operator-5c68f7f4f8-lqzcm    1/1    Running 0    2m51s
community-operators-catalog-58599d96cd-vpj2v 1/1    Running 0    2m51s
control-plane-operator-f6b4c8465-4k5dh      1/1    Running 0    4m32s
etcd-0                                       1/1    Running 0    4m13s
hosted-cluster-config-operator-c4776f89f-dt46j 1/1    Running 0    2m51s
ignition-server-7cd8676fc5-hjx29           1/1    Running 0    4m22s
ingress-operator-75484cdc8c-zhdz5           1/2    Running 0    2m51s
konnectivity-agent-c5485c9df-jsm9s          1/1    Running 0    4m13s
konnectivity-server-85dc754888-7z8vm        1/1    Running 0    4m13s
kube-apiserver-db5fb5549-zlvpq              3/3    Running 0    4m13s
kube-controller-manager-5fbf7b7b7b-mrtjj    1/1    Running 0    90s
kube-scheduler-776c59d757-kfhv6            1/1    Running 0    3m12s
machine-approver-c6b947895-lkdbk            1/1    Running 0    4m13s
oauth-openshift-787b87cff6-trvd6           2/2    Running 0    87s
olm-operator-69c4657864-hxwzk               2/2    Running 0    2m50s
openshift-apiserver-67f9d9c5c7-c9bmv        2/2    Running 0    89s
openshift-controller-manager-5899fc8778-q89xh 1/1    Running 0    2m51s
openshift-oauth-apiserver-569c78c4d-568v8    1/1    Running 0    2m52s
packageserver-ddfffb8d7-wlz6l              2/2    Running 0    2m50s
redhat-marketplace-catalog-7dd77d896-jtxkd   1/1    Running 0    2m51s
redhat-operators-catalog-d66b5c965-qwhn7     1/1    Running 0    2m51s

```

1.7.4.4. 创建 InfraEnv

InfraEnv 是启动实时 **ISO** 的主机可以加入为代理的环境。在这种情况下，代理会在与您托管的 **control plane** 相同的命名空间中创建。

1.

要创建 **InfraEnv**，请输入以下命令：

```

export SSH_PUB_KEY=$(cat $HOME/.ssh/id_rsa.pub)

envsubst <<"EOF" | oc apply -f -
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: ${HOSTED_CLUSTER_NAME}
  namespace: ${HOSTED_CONTROL_PLANE_NAMESPACE}
spec:
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: ${SSH_PUB_KEY}
EOF

```

2.

要生成允许虚拟机或裸机作为代理加入的 **live ISO**，请输入以下命令：

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get InfraEnv
${HOSTED_CLUSTER_NAME} -jsonpath="{.status.isoDownloadURL}"
```

1.7.4.5. 添加代理

您可以通过手动将机器配置为使用 **live ISO** 或使用 **Metal3** 来添加代理。

•

要手动添加代理，请按照以下步骤执行：

1.

下载 **live ISO**，并使用它来启动节点（裸机或虚拟机）。在启动时，节点与 **Assisted Service** 通信，并注册为与 **InfraEnv** 相同的命名空间中的代理。

2.

创建每个代理后，您可以选择在 **spec** 中设置 **install_disk_id** 和 **hostname**。然后，批准它以指示代理可供使用。

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agents
```

输出示例

NAME	CLUSTER	APPROVED	ROLE	STAGE
86f7ac75-4fc4-4b36-8130-40fa12602218			auto-assign	
e57a637f-745b-496e-971d-1abbf03341ba			auto-assign	

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} patch agent 86f7ac75-4fc4-4b36-8130-40fa12602218 -p '{"spec": {"installation_disk_id":"/dev/sda","approved":true,"hostname":"worker-0.example.krnl.es"}}' --type merge
```

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} patch agent 23d0c614-2caa-43f5-b7d3-0b3564688baa -p '{"spec": {"installation_disk_id":"/dev/sda","approved":true,"hostname":"worker-1.example.krnl.es"}}' --type merge
```

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agents
```

输出示例

NAME	CLUSTER	APPROVED	ROLE	STAGE
86f7ac75-4fc4-4b36-8130-40fa12602218		true	auto-assign	
e57a637f-745b-496e-971d-1abbf03341ba		true	auto-assign	

要使用 **Metal3** 添加代理，请按照以下步骤操作：

1.

使用 **Assisted Service** 创建自定义 **ISO** 和 **Baremetal Operator** 来执行安装。

重要： 因为 **BareMetalHost** 对象是在裸机 **Operator** 命名空间外创建的，所以您必须将 **Operator** 配置为监视所有命名空间。

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

metal3 pod 在 **openshift-machine-api** 命名空间中重启。

2.

等待 **metal3 pod** 再次就绪：

```
until oc wait -n openshift-machine-api $(oc get pods -n openshift-machine-api -l baremetal.openshift.io/cluster-baremetal-operator=metal3-state -o name) --for condition=containersready --timeout 10s >/dev/null 2>&1 ; do sleep 1 ; done
```

3.

创建 **BareMetalHost** 对象。您需要配置几个启动裸机节点所需的变量。

- **BMC_USERNAME:** 连接到 **BMC** 的用户名。
- **BMC_PASSWORD** : 连接到 **BMC** 的密码。
- **BMC_IP:** **Metal3** 用来连接到 **BMC** 的 **IP**。

- **WORKER_NAME** : **BaremetalHost** 对象的名称 (该值也用作主机名)
- **BOOT_MAC_ADDRESS** : 连接到 **MachineNetwork** 的 **NIC** 的 **MAC** 地址。
- **UUID** : **Redfish UUID**, 通常为 **1**。如果您使用 **sushy-tools**, 这个值是一个较长的 **UUID**。如果您使用 **iDrac**, 则该值为 **System.Embedded.1**。您可能需要与供应商进行检查。
- **Red HatFISH_SCHEME**: 要使用的 **Redfish** 供应商。如果您使用使用标准 **Redfish** 实现的硬件, 您可以将此值设置为 **redfish-virtualmedia**。 **iDRAC** 使用 **idrac-virtualmedia**。 **iLO5** 使用 **ilo5-virtualmedia**。您可能需要与供应商进行检查。
- **Red HatFISH: Redfish** 连接端点。

```
export BMC_USERNAME=$(echo -n "root" | base64 -w0)
export BMC_PASSWORD=$(echo -n "calvin" | base64 -w0)
export BMC_IP="192.168.124.228"
export WORKER_NAME="ocp-worker-0"
export BOOT_MAC_ADDRESS="aa:bb:cc:dd:ee:ff"
export UUID="1"
export REDFISH_SCHEME="redfish-virtualmedia"
export
REDFISH="${REDFISH_SCHEME}://${BMC_IP}/redfish/v1/Systems/${UUID}"
```

4. 按照以下步骤创建 **BareMetalHost** :

- a. 创建 **BMC Secret** :

```
oc apply -f -
apiVersion: v1
data:
  password: ${BMC_PASSWORD}
  username: ${BMC_USERNAME}
kind: Secret
metadata:
  name: ${WORKER_NAME}-bmc-secret
  namespace: ${HOSTED_CONTROL_PLANE_NAMESPACE}
type: Opaque
```

- b. 创建 **BMH** :

注：**infraenvs.agent-install.openshift.io** 标签用于指定用于启动 **BMH** 的 **InfraEnv**。**bmac.agent-install.openshift.io/hostname** 标签用于手动设置主机名。

如果要手动指定安装磁盘，您可以使用 **BMH** 规格中的 **rootDeviceHints**。如果没有提供 **rootDeviceHints**，代理会选择最适合安装要求的安装磁盘。有关 **rootDeviceHints** 的更多信息，请参阅 **BareMetalHost** 文档中的 **rootDeviceHints** 部分。

```
oc apply -f -
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: ${WORKER_NAME}
  namespace: ${HOSTED_CONTROL_PLANE_NAMESPACE}
  labels:
    infraenvs.agent-install.openshift.io: ${HOSTED_CLUSTER_NAME}
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: ${WORKER_NAME}
spec:
  automatedCleaningMode: disabled
  bmc:
    disableCertificateVerification: True
    address: ${REDFISH}
    credentialsName: ${WORKER_NAME}-bmc-secret
    bootMACAddress: ${BOOT_MAC_ADDRESS}
  online: true
```

代理被自动批准。如果没有批准，请确认 **bootMACAddress** 正确。

置备 **BMH**：

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get bmh
```

输出示例

NAME	STATE	CONSUMER	ONLINE	ERROR	AGE
ocp-worker-0	provisioning	true		2m50s	

BMH 最终达到 置备状态 :

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get bmh
```

输出示例

```
NAME          STATE      CONSUMER ONLINE ERROR AGE
ocp-worker-0  provisioned      true      72s
```

provisioned 意味着节点被配置为从 **virtualCD** 正确启动。显示代理需要一些时间 :

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agent
```

输出示例

```
NAME          CLUSTER APPROVED ROLE    STAGE
4dac1ab2-7dd5-4894-a220-6a3473b67ee6      true   auto-assign
```

代理被自动批准。

c.

对其他两个节点重复此过程 :

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agent
```

输出示例

```
NAME          CLUSTER APPROVED ROLE    STAGE
4dac1ab2-7dd5-4894-a220-6a3473b67ee6      true   auto-assign
d9198891-39f4-4930-a679-65fb142b108b      true   auto-assign
```

```
da503cf1-a347-44f2-875c-4960ddb04091 true auto-assign
```

1.7.4.6. 访问托管集群

托管 **control plane** 正在运行，代理已准备好加入托管集群。在代理加入托管集群前，您需要访问托管集群。

1.

输入以下命令生成 **kubeconfig**：

```
hypershift create kubeconfig --namespace ${CLUSTERS_NAMESPACE} --name
${HOSTED_CLUSTER_NAME} > ${HOSTED_CLUSTER_NAME}.kubeconfig
```

如果访问集群，您可以看到您没有任何节点，并且 **ClusterVersion** 正在尝试协调 **Red Hat OpenShift Container Platform** 发行版本：

```
oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig get clusterversion,nodes
```

输出示例

NAME	VERSION	AVAILABLE	PROGRESSING	SINCE
STATUS				
clusterversion.config.openshift.io/version	4.12z: some cluster operators have not yet rolled out	False	True	8m6s

要让集群正在运行，您需要向其添加节点。

1.7.4.7. 扩展 **NodePool** 对象

您可以通过扩展 **NodePool** 对象将节点添加到托管集群。

1.

将 **NodePool** 对象扩展到两个节点：

■

```
oc -n ${CLUSTERS_NAMESPACE} scale nodepool ${NODEPOOL_NAME} --replicas 2
```

ClusterAPI 代理供应商随机选择两个代理，然后分配给托管集群。这些代理通过不同的状态，最后将托管集群作为 **OpenShift Container Platform** 节点加入。状态的变化过程是从 **binding** 到 **discovering** 到 **insufficient** 到 **installing** 到 **installing-in-progress** 到 **added-to-existing-cluster**。

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agent
```

输出示例

NAME	CLUSTER	APPROVED	ROLE	STAGE
4dac1ab2-7dd5-4894-a220-6a3473b67ee6	hypercluster1	true	auto-assign	
d9198891-39f4-4930-a679-65fb142b108b		true	auto-assign	
da503cf1-a347-44f2-875c-4960ddb04091	hypercluster1	true	auto-assign	

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agent -o jsonpath='{range .items[*]}BMH: {@.metadata.labels.agent-install\.openshift\.io/bmh} Agent: {@.metadata.name} State: {@.status.debugInfo.state}{"\n"}{end}'
```

```
BMH: ocp-worker-2 Agent: 4dac1ab2-7dd5-4894-a220-6a3473b67ee6 State: binding
BMH: ocp-worker-0 Agent: d9198891-39f4-4930-a679-65fb142b108b State: known-unbound
BMH: ocp-worker-1 Agent: da503cf1-a347-44f2-875c-4960ddb04091 State: insufficient
```

2.

代理到达 **add-to-existing-cluster** 状态后，验证您可以看到 **OpenShift Container Platform** 节点：

```
oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig get nodes
```

输出示例

NAME	STATUS	ROLES	AGE	VERSION
ocp-worker-1	Ready	worker	5m41s	v1.24.0+3882f8f
ocp-worker-2	Ready	worker	6m3s	v1.24.0+3882f8f

clusteroperators 开始通过将工作负载添加到节点来协调。您还可以看到扩展 **NodePool** 对

象时创建了两台机器：

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get machines
```

输出示例

NAME	CLUSTER	NODENAME	PROVIDERID
PHASE	AGE	VERSION	
hypercluster1-c96b6f675-m5vch	hypercluster1-b2qhl	ocp-worker-1	agent://da503cf1-a347-44f2-875c-4960ddb04091
Running	15m	4.12z	
hypercluster1-c96b6f675-tl42p	hypercluster1-b2qhl	ocp-worker-2	agent://4dac1ab2-7dd5-4894-a220-6a3473b67ee6
Running	15m	4.12z	

clusterversion 协调最终到达缺少 **Ingress** 和 **Console** 集群 **Operator** 的点：

```
oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig get clusterversion,co
```

NAME	VERSION	AVAILABLE	PROGRESSING	SINCE
STATUS				
clusterversion.config.openshift.io/version	False	True	40m	Unable to apply
4.12z: the cluster operator console has not yet successfully rolled out				

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE	MESSAGE
clusteroperator.config.openshift.io/console	4.12z	False	False	False	11m	RouteHealthAvailable: failed to GET route (https://console-openshift-console.apps.hypercluster1.domain.com): Get "https://console-openshift-console.apps.hypercluster1.domain.com": dial tcp 10.19.3.29:443: connect: connection refused
clusteroperator.config.openshift.io/csi-snapshot-controller	4.12z	True	False	False	10m	
clusteroperator.config.openshift.io/dns	4.12z	True	False	False	9m16s	
clusteroperator.config.openshift.io/image-registry	4.12z	True	False	False	9m5s	
clusteroperator.config.openshift.io/ingress	4.12z	True	False	True	39m	The "default" ingress controller reports Degraded=True: DegradedConditions: One or more other status conditions indicate a degraded state: CanaryChecksSucceeding=False (CanaryChecksRepetitiveFailures: Canary route checks for the default ingress controller are failing)
clusteroperator.config.openshift.io/insights	4.12z	True	False	False	11m	
clusteroperator.config.openshift.io/kube-apiserver	4.12z	True	False	False	40m	
clusteroperator.config.openshift.io/kube-controller-manager	4.12z	True	True	False		

False	False	40m			
clusteroperator.config.openshift.io/kube-scheduler			4.12z	True	False
False	40m				
clusteroperator.config.openshift.io/kube-storage-version-migrator			4.12z	True	
False	False	10m			
clusteroperator.config.openshift.io/monitoring			4.12z	True	False
False	7m38s				
clusteroperator.config.openshift.io/network			4.12z	True	False
False	11m				
clusteroperator.config.openshift.io/openshift-apiserver			4.12z	True	False
False	40m				
clusteroperator.config.openshift.io/openshift-controller-manager			4.12z	True	
False	False	40m			
clusteroperator.config.openshift.io/openshift-samples			4.12z	True	False
False	8m54s				
clusteroperator.config.openshift.io/operator-lifecycle-manager			4.12z	True	
False	False	40m			
clusteroperator.config.openshift.io/operator-lifecycle-manager-catalog			4.12z	True	
False	False	40m			
clusteroperator.config.openshift.io/operator-lifecycle-manager-packageserver			4.12z	True	
False	False	40m			
clusteroperator.config.openshift.io/service-ca			4.12z	True	False
False	11m				
clusteroperator.config.openshift.io/storage			4.12z	True	False
False	11m				

1.7.4.8. 处理入口

每个 **OpenShift Container Platform** 集群都会设置有一个默认的应用程序入口控制器，该控制器预期关联有外部 **DNS** 记录。例如，如果您创建一个名为 **example** 的 **HyperShift** 集群，其基域为 **krnl.es**，您可以预期通配符域 ***.apps.example.krnl.es** 可以被路由。

您可以为 **apps** 设置负载均衡器和通配符 **DNS** 记录。此过程需要部署 **MetalLB**，配置路由到入口部署的新负载均衡器服务，并将通配符 **DNS** 条目分配给负载均衡器 **IP** 地址。

1.

设置 **MetalLB**，以便在创建 **LoadBalancer** 类型的服务时，**MetalLB** 为该服务添加外部 **IP** 地址。

```
cat <<"EOF" | oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig apply -f -
---
apiVersion: v1
kind: Namespace
metadata:
  name: metallb
  labels:
    openshift.io/cluster-monitoring: "true"
  annotations:
    workload.openshift.io/allowed: management
---
```

```

apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: metallb-operator-operatorgroup
  namespace: metallb
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: metallb-operator
  namespace: metallb
spec:
  channel: "stable"
  name: metallb-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace

```

2.

Operator 运行后，创建 **MetalLB** 实例：

```

cat <<"EOF" | oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig apply -f -
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb
EOF

```

3.

通过创建具有单个 **IP** 地址的 **IPAddressPool** 来配置 **MetalLB Operator**。此 **IP** 地址必须与集群节点使用的网络位于同一个子网中。

重要： 更改 **INGRESS_IP** 环境变量以匹配您的环境地址。

```

export INGRESS_IP=192.168.122.23

envsubst <<"EOF" | oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig apply -f -
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: ingress-public-ip
  namespace: metallb
spec:
  protocol: layer2
  autoAssign: false
  addresses:
    - ${INGRESS_IP}-${INGRESS_IP}
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: ingress-public-ip

```

```
namespace: metallb
spec:
  ipAddressPools:
  - ingress-public-ip
EOF
```

4.

按照以下步骤，通过 **MetalLB** 公开 **OpenShift Container Platform Router**：

a.

设置将入口流量路由到入口部署的 **LoadBalancer** 服务。

```
cat <<"EOF" | oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig apply -f -
kind: Service
apiVersion: v1
metadata:
  annotations:
    metallb.universe.tf/address-pool: ingress-public-ip
  name: metallb-ingress
  namespace: openshift-ingress
spec:
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 80
  - name: https
    protocol: TCP
    port: 443
    targetPort: 443
  selector:
    ingresscontroller.operator.openshift.io/deployment-ingresscontroller: default
  type: LoadBalancer
EOF
```

b.

输入以下命令访问 **OpenShift Container Platform** 控制台：

```
curl -kI https://console-openshift-console.apps.example.krn1.es
HTTP/1.1 200 OK
```

c.

检查 **clusterversion** 和 **clusteroperator** 值，以验证一切是否正在运行。输入以下命令：

```
oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig get clusterversion,co
```

输出示例

NAME	VERSION	AVAILABLE	PROGRESSING	SINCE	STATUS
clusterversion.config.openshift.io/version	4.12z	True	False	3m32s	Cluster version is 4.12z

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE	MESSAGE
clusteroperator.config.openshift.io/console	4.12z	True	False	False	3m50s	
clusteroperator.config.openshift.io/csi-snapshot-controller	4.12z	True	False	False	25m	
clusteroperator.config.openshift.io/dns	4.12z	True	False	False	23m	
clusteroperator.config.openshift.io/image-registry	4.12z	True	False	False	23m	
clusteroperator.config.openshift.io/ingress	4.12z	True	False	False	53m	
clusteroperator.config.openshift.io/insights	4.12z	True	False	False	25m	
clusteroperator.config.openshift.io/kube-apiserver	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/kube-controller-manager	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/kube-scheduler	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/kube-storage-version-migrator	4.12z	True	False	False	25m	
clusteroperator.config.openshift.io/monitoring	4.12z	True	False	False	21m	
clusteroperator.config.openshift.io/network	4.12z	True	False	False	25m	
clusteroperator.config.openshift.io/openshift-apiserver	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/openshift-controller-manager	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/openshift-samples	4.12z	True	False	False	23m	
clusteroperator.config.openshift.io/operator-lifecycle-manager	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/operator-lifecycle-manager-catalog	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/operator-lifecycle-manager-packageserver	4.12z	True	False	False	54m	
clusteroperator.config.openshift.io/service-ca	4.12z	True	False	False	25m	
clusteroperator.config.openshift.io/storage	4.12z	True	False	False	25m	

1.7.4.9. 为托管集群启用节点自动扩展

当托管集群和备用代理中需要更多容量时，您可以启用自动扩展来安装新代理。

1.

要启用自动扩展，请输入以下命令。在这种情况下，最少的节点数量为 **2**，最大数量为 **5**。

```
oc -n ${CLUSTERS_NAMESPACE} patch nodepool ${HOSTED_CLUSTER_NAME} --
type=json -p '[{"op": "remove", "path": "/spec/replicas"}, {"op": "add", "path":
"/spec/autoScaling", "value": { "max": 5, "min": 2 }}]'
```

如果 **10** 分钟通过不需要额外容量，则代理将停用，并再次放在备用队列中。

2.

创建需要新节点的工作负载。

```
cat <<EOF | oc --kubeconfig ${HOSTED_CLUSTER_NAME}.kubeconfig apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: reversewords
  name: reversewords
  namespace: default
spec:
  replicas: 40
  selector:
    matchLabels:
      app: reversewords
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: reversewords
    spec:
      containers:
      - image: quay.io/mavazque/reversewords:latest
        name: reversewords
        resources:
          requests:
            memory: 2Gi
    status: {}
EOF
```

3.

输入以下命令验证剩余的代理是否已部署。在本例中，置备备用代理 **d9198891-39f4-4930-a679-65fb142b108b**：

```
oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agent -o jsonpath='{range
.items[*]}BMH: {@.metadata.labels.agent-install\.openshift\.io/bmh} Agent:
{@.metadata.name} State: {@.status.debugInfo.state}{"\n"}{end}'
```

输出示例

```
BMH: ocp-worker-2 Agent: 4dac1ab2-7dd5-4894-a220-6a3473b67ee6 State: added-to-
existing-cluster
BMH: ocp-worker-0 Agent: d9198891-39f4-4930-a679-65fb142b108b State: installing-in-
progress
BMH: ocp-worker-1 Agent: da503cf1-a347-44f2-875c-4960ddb04091 State: added-to-
existing-cluster
```

4.

如果输入以下命令来检查节点，则输出中会显示新节点。在本例中，**ocp-worker-0** 添加到集群中：

```
oc --kubecfg ${HOSTED_CLUSTER_NAME}.kubecfg get nodes
```

输出示例

NAME	STATUS	ROLES	AGE	VERSION
ocp-worker-0	Ready	worker	35s	v1.24.0+3882f8f
ocp-worker-1	Ready	worker	40m	v1.24.0+3882f8f
ocp-worker-2	Ready	worker	41m	v1.24.0+3882f8f

5.

要删除节点，请输入以下命令删除工作负载：

```
oc --kubecfg ${HOSTED_CLUSTER_NAME}.kubecfg -n default delete deployment
reversewords
```

6.

输入以下命令等待 **10** 分钟，并确认节点已被删除：

```
oc --kubecfg ${HOSTED_CLUSTER_NAME}.kubecfg get nodes
```

输出示例

```

NAME          STATUS  ROLES  AGE  VERSION
ocp-worker-1  Ready  worker  51m  v1.24.0+3882f8f
ocp-worker-2  Ready  worker  52m  v1.24.0+3882f8f

```

```

oc -n ${HOSTED_CONTROL_PLANE_NAMESPACE} get agent -o jsonpath='{range
.items[*]}BMH: {@.metadata.labels.agent-install\.openshift\.io/bmh} Agent:
{@.metadata.name} State: {@.status.debugInfo.state}{"\n"}{end}'

```

```

BMH: ocp-worker-2 Agent: 4dac1ab2-7dd5-4894-a220-6a3473b67ee6 State: added-to-
existing-cluster

```

```

BMH: ocp-worker-0 Agent: d9198891-39f4-4930-a679-65fb142b108b State: known-unbound

```

```

BMH: ocp-worker-1 Agent: da503cf1-a347-44f2-875c-4960ddb04091 State: added-to-
existing-cluster

```

1.7.4.10. 验证托管集群创建

部署过程完成后，您可以验证托管集群是否已成功创建。在创建托管集群后，按照以下步骤操作。

1.

输入 **extract** 命令获取新托管集群的 **kubeconfig**：

```

oc extract -n kni21 secret/kni21-admin-kubeconfig --to=- > kubeconfig-kni21
# kubeconfig

```

2.

使用 **kubeconfig** 查看托管集群的集群 **Operator**。输入以下命令：

```

oc get co --kubeconfig=kubeconfig-kni21

```

输出示例

```

NAME          VERSION AVAILABLE PROGRESSING DEGRADED
SINCE MESSAGE
console       4.10.26 True      False      False      2m38s
csi-snapshot-controller 4.10.26 True      False      False      4m3s
dns           4.10.26 True      False      False      2m52s
image-registry 4.10.26 True      False      False      2m8s
ingress       4.10.26 True      False      False      22m

```

kube-apiserver	4.10.26	True	False	False	23m
kube-controller-manager	4.10.26	True	False	False	23m
kube-scheduler	4.10.26	True	False	False	23m
kube-storage-version-migrator	4.10.26	True	False	False	4m52s
monitoring	4.10.26	True	False	False	69s
network	4.10.26	True	False	False	4m3s
node-tuning	4.10.26	True	False	False	2m22s
openshift-apiserver	4.10.26	True	False	False	23m
openshift-controller-manager	4.10.26	True	False	False	23m
openshift-samples	4.10.26	True	False	False	2m15s
operator-lifecycle-manager	4.10.26	True	False	False	22m
operator-lifecycle-manager-catalog	4.10.26	True	False	False	23m
operator-lifecycle-manager-packageserver	4.10.26	True	False	False	23m
service-ca	4.10.26	True	False	False	4m41s
storage	4.10.26	True	False	False	4m43s

3.

您还可以输入以下命令查看托管集群中运行的 **pod** :

```
oc get pods -A --kubeconfig=kubeconfig-kni21
```

输出示例

NAMESPACE	STATUS	RESTARTS	AGE	NAME	READY
kube-system	Running	0	3m52s	konnnectivity-agent-khlqv	0/1
kube-system	Running	0	4m24s	konnnectivity-agent-nrbvw	0/1
kube-system	Running	0	4m14s	konnnectivity-agent-s5p7g	0/1
kube-system	1/1 Running	0	5m56s	kube-apiserver-proxy-asus3-vm1.kni.schmaustech.com	
kube-system	1/1 Running	0	6m37s	kube-apiserver-proxy-asus3-vm2.kni.schmaustech.com	
kube-system	1/1 Running	0	6m17s	kube-apiserver-proxy-asus3-vm3.kni.schmaustech.com	
openshift-cluster-node-tuning-operator	9cf2k	1/1 Running	0	cluster-node-tuning-operator-798fcd89dc-20m	
openshift-cluster-node-tuning-operator	Running	0	109s	tuned-dhw5p	1/1
openshift-cluster-node-tuning-operator	Running	0	110s	tuned-dlp8f	1/1
openshift-cluster-node-tuning-operator	Running	0	109s	tuned-l569k	1/1
openshift-cluster-samples-operator	2/2 Running	0	20m	cluster-samples-operator-6b5bcb9dff-kpnbc	
openshift-cluster-storage-operator				cluster-storage-operator-5f784969f5-vwzgz	

1/1	Running	1 (113s ago)	20m		
				openshift-cluster-storage-operator	csi-snapshot-controller-6b7687b7d9-7nrfw
1/1	Running	0	3m8s		
				openshift-cluster-storage-operator	csi-snapshot-controller-6b7687b7d9-csksg
1/1	Running	0	3m9s		
				openshift-cluster-storage-operator	csi-snapshot-controller-operator-7f4d9fc5b8-
hkvrk	1/1	Running	0	20m	
				openshift-cluster-storage-operator	csi-snapshot-webhook-6759b5dc8b-7qltn
1/1	Running	0	3m12s		
				openshift-cluster-storage-operator	csi-snapshot-webhook-6759b5dc8b-f8bqk
1/1	Running	0	3m12s		
				openshift-console-operator	console-operator-8675b58c4c-flc5p
1/1	Running	1 (96s ago)	20m		
				openshift-console	console-5cbf6c7969-6gk6z
Running	0	119s			1/1
				openshift-console	downloads-7bcd756565-6wj5j
Running	0	4m3s			1/1
				openshift-dns-operator	dns-operator-77d755cd8c-xjfbn
Running	0	21m			2/2
				openshift-dns	dns-default-jwjkz
Running	0	113s			2/2
				openshift-dns	dns-default-kfqnh
Running	0	113s			2/2
				openshift-dns	dns-default-xlqsm
Running	0	113s			2/2
				openshift-dns	node-resolver-jzxnd
Running	0	110s			1/1
				openshift-dns	node-resolver-xqdr5
Running	0	110s			1/1
				openshift-dns	node-resolver-zl6h4
Running	0	110s			1/1
				openshift-image-registry	cluster-image-registry-operator-64fcdfbf5-r7d5t
1/1	Running	0	20m		
				openshift-image-registry	image-registry-7dfd99d68-t9ppq9
1/1	Running	0	53s		
				openshift-image-registry	node-ca-hkfnr
Running	0	56s			1/1
				openshift-image-registry	node-ca-vlsdl
Running	0	56s			1/1
				openshift-image-registry	node-ca-xqnsd
Running	0	56s			1/1
				openshift-ingress-canary	ingress-canary-86z6r
Running	0	4m13s			1/1
				openshift-ingress-canary	ingress-canary-8jhxc
Running	0	3m52s			1/1
				openshift-ingress-canary	ingress-canary-cv45h
Running	0	4m24s			1/1
				openshift-ingress	router-default-6bb8944f66-z2lxr
Running	0	20m			1/1
				openshift-kube-storage-version-migrator-operator	kube-storage-version-migrator-operator-
56b57b4844-p9zgp	1/1	Running	1 (2m16s ago)	20m	
				openshift-kube-storage-version-migrator	migrator-58bb4d89d5-5sl9w
1/1	Running	0	3m30s		
				openshift-monitoring	alertmanager-main-0
Running	0	100s			6/6
				openshift-monitoring	cluster-monitoring-operator-5bc5885cd4-dwbc4

2/2	Running	0	20m		
	openshift-monitoring			grafana-78f798868c-wd84p	3/3
	Running	0	94s		
	openshift-monitoring			kube-state-metrics-58b8f97f6c-6kp4v	
3/3	Running	0	104s		
	openshift-monitoring			node-exporter-ll7cp	2/2
	Running	0	103s		
	openshift-monitoring			node-exporter-tgsqg	2/2
	Running	0	103s		
	openshift-monitoring			node-exporter-z99gr	2/2
	Running	0	103s		
	openshift-monitoring			openshift-state-metrics-677b9fb74f-qqp6g	
3/3	Running	0	104s		
	openshift-monitoring			prometheus-adapter-f69fff5f9-7tdn9	0/1
	Running	0	17s		
	openshift-monitoring			prometheus-k8s-0	6/6
	Running	0	93s		
	openshift-monitoring			prometheus-operator-6b9d4fd9bd-tqfcx	
2/2	Running	0	2m2s		
	openshift-monitoring			telemeter-client-74d599658c-wqw5j	
3/3	Running	0	101s		
	openshift-monitoring			thanos-querier-64c8757854-z4lll	6/6
	Running	0	98s		
	openshift-multus			multus-additional-cni-plugins-cqst9	1/1
	Running	0	6m14s		
	openshift-multus			multus-additional-cni-plugins-dbmkj	1/1
	Running	0	5m56s		
	openshift-multus			multus-additional-cni-plugins-kcw19	1/1
	Running	0	6m14s		
	openshift-multus			multus-admission-controller-22cmb	2/2
	Running	0	3m52s		
	openshift-multus			multus-admission-controller-256tn	2/2
	Running	0	4m13s		
	openshift-multus			multus-admission-controller-mz9jm	2/2
	Running	0	4m24s		
	openshift-multus			multus-bxgvr	1/1
	Running	0	6m14s		
	openshift-multus			multus-dmkdc	1/1
	Running	0	6m14s		
	openshift-multus			multus-gqw2f	1/1
	Running	0	5m56s		
	openshift-multus			network-metrics-daemon-6cx4x	2/2
	Running	0	5m56s		
	openshift-multus			network-metrics-daemon-gz4jp	2/2
	Running	0	6m13s		
	openshift-multus			network-metrics-daemon-jq9j4	2/2
	Running	0	6m13s		
	openshift-network-diagnostics			network-check-source-8497dc8f86-cn4nm	
1/1	Running	0	5m59s		
	openshift-network-diagnostics			network-check-target-d8db9	
1/1	Running	0	5m58s		
	openshift-network-diagnostics			network-check-target-jdbv8	
1/1	Running	0	5m58s		
	openshift-network-diagnostics			network-check-target-zzmdv	
1/1	Running	0	5m55s		
	openshift-network-operator			network-operator-f5b48cd67-x5dcz	

1/1	Running	0	21m		
openshift-sdn			sdn-452r2	2/2	Running
0	5m56s				
openshift-sdn			sdn-68g69	2/2	Running
0	6m				
openshift-sdn			sdn-controller-4v5mv	2/2	
Running	0	5m56s			
openshift-sdn			sdn-controller-crsc	2/2	
Running	0	6m1s			
openshift-sdn			sdn-controller-fxtn9	2/2	
Running	0	6m1s			
openshift-sdn			sdn-n5jm5	2/2	Running
0	6m				
openshift-service-ca-operator			service-ca-operator-5bf7f9d958-vnqcg		
1/1	Running	1 (2m ago)	20m		
openshift-service-ca			service-ca-6c54d7944b-v5mrw		1/1
Running	0	3m8s			

1.7.4.11. 在裸机上销毁托管集群

您可以使用控制台销毁裸机托管集群。完成以下步骤以在裸机上销毁托管集群：

1. 在控制台中，进入到 **Infrastructure > Clusters**。
2. 在 **Clusters** 页面中，选择要销毁的集群。
3. 在 **Actions** 菜单中，选择 **Destroy** 集群 来删除集群。

1.7.4.12. 其他资源

- 如需有关 **MetalLB** 的更多信息，请参阅 **OpenShift Container Platform** 文档中的 [关于 MetalLB 和 MetalLB Operator](#)。
- 有关 **rootDeviceHints** 的更多信息，请参阅 **BareMetalHost** 文档中的 [rootDeviceHints](#) 部分。

1.7.5. 禁用托管的 control plane 功能

您可以卸载 **HyperShift Operator** 并禁用托管的 **control plane**。禁用托管的 **control plane** 集群功能时，您必须销毁托管集群以及 **multicluster engine operator** 上的受管集群资源，如 *管理托管的 control plane 集群* 主题 中所述。

1.7.5.1. 卸载 HyperShift Operator

要卸载 **HyperShift Operator** 并从 **local-cluster** 禁用 **hypershift-addon**，请完成以下步骤：

1. 运行以下命令，以确保没有托管集群正在运行：

```
oc get hostedcluster -A
```

重要：如果托管集群正在运行，则 **HyperShift Operator** 不会卸载，即使 **hypershift-addon** 被禁用。

2. 运行以下命令来禁用 **hypershift-addon**：

```
oc patch mce multiclusterengine --type=merge -p '{"spec":{"overrides":{"components":[{"name":"hypershift-local-hosting","enabled": false}]}}}'
```

提示：在禁用 **hypershift-addon** 后，您还可以从 **multicluster engine operator** 控制台为 **local-cluster** 禁用 **hypershift-addon**。

1.7.5.2. 禁用托管的 control plane 功能

在禁用托管的 **control plane** 功能前，您必须首先卸载 **HyperShift Operator**。运行以下命令来禁用托管的 **control plane** 功能：

```
oc patch mce multiclusterengine --type=merge -p '{"spec":{"overrides":{"components":[{"name":"hypershift-preview","enabled": false}]}}}'
```

您可以运行以下命令来验证 **MultiClusterEngine** 自定义资源中禁用了 **hypershift-preview** 和 **hypershift-local-hosting** 功能：

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

请参阅以下示例，其中 **hypershift-preview** 和 **hypershift-local-hosting** 的 **enabled:** 标记设为 **false**：

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

1.7.5.3. 其他资源

- [在 AWS 上管理托管的 control plane](#)
- [管理托管的 control plane 集群](#)

1.8. API

您可以使用 **multicluster engine operator** 访问集群生命周期管理的以下 **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 \(v1beta2\)](#)
- [Clusterview API](#)
- [ClusterSetBindings API \(v1beta2\)](#)
- [MultiClusterEngine API](#)
- [Placements API \(v1beta1\)](#)

- [PlacementDecisions API \(v1beta1\)](#)
- [管理的服务帐户（技术预览）](#)

1.8.1. Clusters API

1.8.1.1. 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**。

1.8.1.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.8.1.1.2. Tags

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

1.8.1.2. 路径

1.8.1.2.1. 查询所有集群

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

1.8.1.2.1.1. 描述

查询集群以获取更多详细信息。

1.8.1.2.1.2. 参数

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

1.8.1.2.1.3. 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

1.8.1.2.1.4. 使用

- `cluster/yaml`

1.8.1.2.1.5. Tags

- `cluster.open-cluster-management.io`

1.8.1.2.2. 创建集群

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

1.8.1.2.2.1. 描述

创建集群

1.8.1.2.2.2. 参数

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

1.8.1.2.2.3. 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

1.8.1.2.2.4. 使用

- `cluster/yaml`

1.8.1.2.2.5. Tags

- `cluster.open-cluster-management.io`

1.8.1.2.2.6. HTTP 请求示例

1.8.1.2.2.6.1. 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": {}
}
```

1.8.1.2.3. 查询单个集群

GET /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}**1.8.1.2.3.1. 描述**

查询单个集群以获取更多详细信息。

1.8.1.2.3.2. 参数

类型	Name	描述	模式
标头	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

1.8.1.2.3.3. 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

1.8.1.2.3.4. Tags

- **cluster.open-cluster-management.io**

1.8.1.2.4. 删除集群**DELETE /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}****1.8.1.2.4.1. 描述**

删除单个集群

1.8.1.2.4.2. 参数

类型	Name	描述	模式
标头	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

1.8.1.2.4.3. 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

1.8.1.2.4.4. Tags

- **cluster.open-cluster-management.io**

1.8.1.3. 定义

1.8.1.3.1. 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)

1.8.2. ClusterSets API (v1beta2)**1.8.2.1. 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**。

1.8.2.1.1. URI scheme

BasePath : /kubernetes/apis

Schemes : HTTPS

1.8.2.1.2. Tags

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

1.8.2.2. 路径

1.8.2.2.1. 查询所有集群集 (clusterset)

GET /cluster.open-cluster-management.io/v1beta2/managedclustersets

1.8.2.2.1.1. 描述

查询 **Clustersets** 以获取更多详细信息。

1.8.2.2.1.2. 参数

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

1.8.2.2.1.3. 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

1.8.2.2.1.4. 使用

- **clusterset/yaml**

1.8.2.2.1.5. Tags

- **cluster.open-cluster-management.io**

1.8.2.2.2. 创建一个 clusterset

POST /cluster.open-cluster-management.io/v1beta2/managedclustersets

1.8.2.2.2.1. 描述

创建 **Clusterset**。

1.8.2.2.2.2. 参数

类型	Name	描述	模式
标头	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

1.8.2.2.2.3. 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

1.8.2.2.2.4. 使用

- **clusterset/yaml**

1.8.2.2.2.5. Tags

- **cluster.open-cluster-management.io**

1.8.2.2.2.6. HTTP 请求示例

1.8.2.2.2.6.1. 请求正文

```
{
  "apiVersion": "cluster.open-cluster-management.io/v1beta2",
  "kind": "ManagedClusterSet",
  "metadata": {
    "name": "clusterset1"
  },
  "spec": { },
  "status": { }
}
```

1.8.2.2.3. 查询单个集群集

```
GET /cluster.open-cluster-management.io/v1beta2/managedclustersets/{clusterset_name}
```

1.8.2.2.3.1. 描述

查询单个集群集以获取更多详细信息。

1.8.2.2.3.2. 参数

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

1.8.2.2.3.3. 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

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

1.8.2.2.3.4. Tags

- **cluster.open-cluster-management.io**

1.8.2.2.4. 删除集群集

DELETE /cluster.open-cluster-management.io/v1beta2/managedclustersets/{clusterset_name}

1.8.2.2.4.1. 描述

删除单个集群集。

1.8.2.2.4.2. 参数

类型	Name	描述	模式
标头	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

1.8.2.2.4.3. 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

HTTP 代码	描述	模式
503	服务不可用	无内容

1.8.2.2.4.4. Tags

- `cluster.open-cluster-management.io`

1.8.2.3. 定义

1.8.2.3.1. Clusterset

Name	模式
<code>apiVersion</code> <i>必需</i>	字符串
<code>kind</code> <i>必需</i>	字符串
<code>metadata</code> <i>必需</i>	对象

1.8.3. Clustersetbindings API (v1beta2)

1.8.3.1. 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`。

1.8.3.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.8.3.1.2. Tags

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

1.8.3.2. 路径

1.8.3.2.1. 查询所有 clustersetbindings

GET /cluster.open-cluster-management.io/v1beta2/namespaces/{namespace}/managedclustersetbindings

1.8.3.2.1.1. 描述

查询 **clustersetbindings** 以获取更多详细信息。

1.8.3.2.1.2. 参数

类型	Name	描述	模式
标头	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

1.8.3.2.1.3. 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

1.8.3.2.1.4. 使用

- **clustersetbinding/yaml**

1.8.3.2.1.5. Tags

- **cluster.open-cluster-management.io**

1.8.3.2.2. 创建 clustersetbinding

POST /cluster.open-cluster-management.io/v1beta2/namespaces/{namespace}/managedclustersetbindings

1.8.3.2.2.1. 描述

创建 `clustersetbinding`。

1.8.3.2.2.2. 参数

类型	Name	描述	模式
标头	COOKIE <i>必需</i>	身份验证 : Bearer {ACCESS_TOKEN}; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	namespace <i>必需</i>	要使用的命名空间, 如 default。	字符串
Body	body <i>必需</i>	描述要创建的 <code>clustersetbinding</code> 的参数。	Clustersetbinding

1.8.3.2.2.3. 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

1.8.3.2.2.4. 使用

- `clustersetbinding/yaml`

1.8.3.2.2.5. Tags

- `cluster.open-cluster-management.io`

1.8.3.2.2.6. HTTP 请求示例

1.8.3.2.2.6.1. 请求正文

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

1.8.3.2.3. 查询单个 clustersetbinding

```
GET /cluster.open-cluster-management.io/v1beta2/namespaces/{namespace}/managedclustersetbindings/{clustersetbinding_name}
```

1.8.3.2.3.1. 描述

查询单个 **clustersetbinding** 获取更多详细信息。

1.8.3.2.3.2. 参数

类型	Name	描述	模式
标头	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

1.8.3.2.3.3. 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

1.8.3.2.3.4. Tags

- `cluster.open-cluster-management.io`

1.8.3.2.4. 删除 clustersetbinding

DELETE /cluster.open-cluster-management.io/v1beta2/managedclustersetbindings/{clustersetbinding_name}

1.8.3.2.4.1. 描述

删除单个 clustersetbinding。

1.8.3.2.4.2. 参数

类型	Name	描述	模式
标头	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

1.8.3.2.4.3. 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

1.8.3.2.4.4. Tags

- `cluster.open-cluster-management.io`

1.8.3.3. 定义

1.8.3.3.1. Clustersetbinding

Name	模式
<code>apiVersion</code> <i>必需</i>	字符串
<code>kind</code> <i>必需</i>	字符串
<code>metadata</code> <i>必需</i>	对象
<code>spec</code> <i>必需</i>	<code>spec</code>

`spec`

Name	模式
<code>clusterSet</code> <i>必需</i>	字符串

1.8.4. Clusterview API (v1alpha1)

1.8.4.1. 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**。

1.8.4.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.8.4.1.2. Tags

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

1.8.4.2. 路径

1.8.4.2.1. 获取受管集群

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

1.8.4.2.1.1. 描述

查看您可以访问的受管集群列表。

1.8.4.2.1.2. 参数

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

1.8.4.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容

HTTP 代码	描述	模式
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

1.8.4.2.1.4. 使用

- `managedcluster/yaml`

1.8.4.2.1.5. Tags

- `clusterview.open-cluster-management.io`

1.8.4.2.2. 列出受管集群

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

1.8.4.2.2.1. 描述

查看您可以访问的受管集群列表。

1.8.4.2.2.2. 参数

类型	Name	描述	模式
标头	COOKIE <i>必需</i>	身份验证 : Bearer {ACCESS_TOKEN}; ACCESS_TOKEN 是用户访问令牌。	字符串
Body	body <i>可选</i>	要列出受管集群的用户 ID 的名称。	字符串

1.8.4.2.2.3. 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

1.8.4.2.2.4. 使用

- `managedcluster/yaml`

1.8.4.2.2.5. Tags

- `clusterview.open-cluster-management.io`

1.8.4.2.2.6. HTTP 请求示例

1.8.4.2.2.6.1. 请求正文

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

1.8.4.2.3. 观察受管集群集

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

1.8.4.2.3.1. 描述

观察您可以访问的受管集群。

1.8.4.2.3.2. 参数

类型	Name	描述	模式
标头	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

1.8.4.2.3.3. 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

1.8.4.2.4. 列出受管集群集。

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

1.8.4.2.4.1. 描述

列出您可以访问的受管集群。

1.8.4.2.4.2. 参数

类型	Name	描述	模式
标头	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

1.8.4.2.4.3. 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

1.8.4.2.5. 列出受管集群集。

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

1.8.4.2.5.1. 描述

列出您可以访问的受管集群。

1.8.4.2.5.2. 参数

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

1.8.4.2.5.3. 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

1.8.4.2.6. 观察受管集群集。

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

1.8.4.2.6.1. 描述

观察您可以访问的受管集群。

1.8.4.2.6.2. 参数

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

1.8.4.2.6.3. 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

1.8.5. 管理的服务帐户（技术预览）

1.8.5.1. 概述

本文档介绍了与 **multicluster engine operator** 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**。

1.8.5.1.1. URI scheme

BasePath : /kubernetes/apis

Schemes : HTTPS

1.8.5.1.2. Tags

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

1.8.5.2. 路径

1.8.5.2.1. 创建 **ManagedServiceAccount**

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

1.8.5.2.1.1. 描述

创建 **ManagedServiceAccount**。

1.8.5.2.1.2. 参数

类型	Name	描述	模式
标头	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

1.8.5.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容

HTTP 代码	描述	模式
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

1.8.5.2.1.4. 使用

- `managedserviceaccount/yaml`

1.8.5.2.1.5. Tags

- `managedserviceaccount.multicluster.openshift.io`

1.8.5.2.1.5.1. 请求正文

```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "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 object
represents. Servers may infer this from the endpoint the client submits 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 will be
rotated from the upstream",
          "type": "boolean"
        },
        "validity": {
          "default": "8640h0m0s",
          "description": "Validity is the duration for which the signed
ServiceAccount token is valid.",
          "type": "string"
        }
      },
      "type": "object"
    },
    "ttlSecondsAfterCreation": {
      "description": "ttlSecondsAfterCreation limits the lifetime of a
ManagedServiceAccount. If the ttlSecondsAfterCreation field is set, the
ManagedServiceAccount will be automatically deleted regardless of the
ManagedServiceAccount's status. When the ManagedServiceAccount is deleted, its
lifecycle guarantees (e.g. finalizers) will be honored. If this field is unset, the
ManagedServiceAccount won't be automatically deleted. If this field is set to zero, the
ManagedServiceAccount becomes eligible for deletion immediately after its creation. In
order to use ttlSecondsAfterCreation, the 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,\n\n type FooStatus struct{ // Represents the observations of
a\nfoo's current state. // Known .status.conditions.type are:\n\n\"Available\", \"Progressing\",
and \"Degraded\" // +patchMergeKey=type\n // +patchStrategy=merge // +listType=map
// +listMapKey=type\n\n Conditions []metav1.Condition
`json:\n\"conditions,omitempty\"\n\npatchStrategy:\n\"merge\" patchMergeKey:\n\"type\"
protobuf:\n\"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 certificate 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": []
}
}

```

1.8.5.2.2. 查询单个 ManagedServiceAccount

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

1.8.5.2.2.1. 描述

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

1.8.5.2.2.2. 参数

类型	Name	描述	模式
标头	COOKIE <i>必需</i>	身份验证 : Bearer {ACCESS_TOKEN}; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	managedserviceaccount_name <i>required</i>	要查询的 ManagedServiceAccount 的名称。	字符串

1.8.5.2.2.3. 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

1.8.5.2.2.4. Tags

- `cluster.open-cluster-management.io`

1.8.5.2.3. 删除 ManagedServiceAccount

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

1.8.5.2.3.1. 描述

删除单个 **ManagedServiceAccount**。

1.8.5.2.3.2. 参数

类型	Name	描述	模式
标头	COOKIE <i>必需</i>	身份验证 : Bearer {ACCESS_TOKEN}; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	managedserviceaccount_name <i>required</i>	要删除的 ManagedServiceAccount 的名称。	字符串

1.8.5.2.3.3. 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

1.8.5.2.3.4. Tags

- cluster.open-cluster-management.io

1.8.5.3. 定义

1.8.5.3.1. ManagedServiceAccount

Name	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。	

1.8.6. MultiClusterEngine API

1.8.6.1. 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**。

1.8.6.1.1. URI scheme

BasePath : /kubernetes/apis

Schemes : HTTPS

1.8.6.1.2. Tags

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

1.8.6.2. 路径

1.8.6.2.1. 创建 MultiClusterEngine

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

1.8.6.2.1.1. 描述

创建一个 **MultiClusterEngine**。

1.8.6.2.1.2. 参数

类型	Name	描述	模式
标头	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

1.8.6.2.1.3. 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

1.8.6.2.1.4. 使用

- **MultiClusterEngines/yaml**

1.8.6.2.1.5. Tags

- **multicusterengines.multicuster.openshift.io**

1.8.6.2.1.5.1. 请求正文

```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "multicusterengines.multicuster.openshift.io"
  },
  "spec": {
    "group": "multicuster.openshift.io",
    "names": {
      "kind": "MultiClusterEngine",
      "listKind": "MultiClusterEngineList",
      "plural": "multicusterengines",
      "shortNames": [
        "mce"
      ],
      "singular": "multicusterengine"
    },
    "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 multicusterengines\nAPI",
            "properties": {
              "apiVersion": {
                "description": "APIVersion defines the versioned schema of this representation\nof"

```

an object. Servers should convert recognized schemas to the latest internal 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 object
represents. Servers may infer this from the endpoint the client submits 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 operand
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 taint that
matches the triple <key,value,effect> using the matching operator <operator>.",
          "properties": {
            "effect": {
              "description": "Effect indicates the taint effect to match. Empty means match
all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and
NoExecute.",
              "type": "string"
            },
            "key": {
              "description": "Key is the taint key that the toleration applies to. Empty
means match all taint keys. If the key is empty, operator must be Exists; this combination
means to match all values and all keys.",
              "type": "string"
            },
            "operator": {
              "description": "Operator represents a key's relationship to the value. Valid
operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value,

```

so that a pod can tolerate all taints of a particular category.",

```

    "type": "string"
  },
  "tolerationSeconds": {
    "description": "TolerationSeconds represents the period of time the toleration
    (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By
    default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative
    values will be treated as 0 (evict immediately) by the system.",
    "format": "int64",
    "type": "integer"
  },
  "value": {
    "description": "Value is the taint value the toleration matches to. If the
    operator is Exists, the value should be empty, otherwise 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 tracked
        components",
        "properties": {
          "kind": {
            "description": "The resource kind this condition represents",
            "type": "string"
          },
          "lastTransitionTime": {
            "description": "LastTransitionTime is the last time the condition changed
            from one status to another.",
            "format": "date-time",
            "type": "string"
          },
          "message": {
            "description": "Message is a human-readable message indicating details
            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 last 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": []
}
}

```

1.8.6.2.2. 查询所有 MultiClusterEngines

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

1.8.6.2.2.1. 描述

查询多集群引擎以获取更多详细信息。

1.8.6.2.2.2. 参数

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

1.8.6.2.2.3. 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

1.8.6.2.2.4. 使用

- `operator/yaml`

1.8.6.2.2.5. Tags

- `multiclusterengines.multicluster.openshift.io`

1.8.6.2.3. 删除 MultiClusterEngine Operator

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

1.8.6.2.3.1. 参数

类型	Name	描述	模式
标头	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

1.8.6.2.3.2. 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

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

1.8.6.2.3.3. Tags

- `multiclusterengines.multicluster.openshift.io`

1.8.6.3. 定义

1.8.6.3.1. MultiClusterEngine

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

1.8.6.3.2. specs 列表

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

1.8.7. Placements API (v1beta1)

1.8.7.1. 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**。

1.8.7.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.8.7.1.2. Tags

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

1.8.7.2. 路径

1.8.7.2.1. 查询所有放置

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

1.8.7.2.1.1. 描述

查询您的放置以获取更多详细信息。

1.8.7.2.1.2. 参数

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

1.8.7.2.1.3. 响应

HTTP 代码	描述	模式
200	成功	无内容
403	禁止访问	无内容

HTTP 代码	描述	模式
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

1.8.7.2.1.4. 使用

- `placement/yaml`

1.8.7.2.1.5. Tags

- `cluster.open-cluster-management.io`

1.8.7.2.2. 创建一个放置

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

1.8.7.2.2.1. 描述

创建一个放置。

1.8.7.2.2.2. 参数

类型	Name	描述	模式
标头	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

1.8.7.2.2.3. 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

1.8.7.2.2.4. 使用

- `placement/yaml`

1.8.7.2.2.5. Tags

- `cluster.open-cluster-management.io`

1.8.7.2.2.6. HTTP 请求示例

1.8.7.2.2.6.1. 请求正文

```
{
  "apiVersion": "cluster.open-cluster-management.io/v1beta1",
  "kind": "Placement",
  "metadata": {
    "name": "placement1",
    "namespace": "ns1"
  },
  "spec": {
    "predicates": [
      {
        "requiredClusterSelector": {
          "labelSelector": {
            "matchLabels": {
              "vendor": "OpenShift"
            }
          }
        }
      }
    ]
  },
  "status": {}
}
```

1.8.7.2.3. 查询单个放置

GET /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/placements/{placement_name}

1.8.7.2.3.1. 描述

查询单个放置以获取更多详细信息。

1.8.7.2.3.2. 参数

类型	Name	描述	模式
标头	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

1.8.7.2.3.3. 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

1.8.7.2.3.4. Tags

- **cluster.open-cluster-management.io**

1.8.7.2.4. 删除一个放置

DELETE /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/placements/{placement_name}

1.8.7.2.4.1. 描述

删除一个放置。

1.8.7.2.4.2. 参数

类型	Name	描述	模式
标头	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

1.8.7.2.4.3. 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

1.8.7.2.4.4. Tags

- **cluster.open-cluster-management.io**

1.8.7.3. 定义

1.8.7.3.1. Placement

Name	描述	模式
apiVersion 必需	放置的版本化模式。	字符串
kind 必需	代表 REST 资源的字符串值。	字符串

Name	描述	模式
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 数组

clusterPredicate

Name	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

Name	描述	模式
labelSelector 可选	按标签的 ManagedClusters 选择器。	对象
claimSelector 可选	按声明的 ManagedClusters 选择器。	clusterClaimSelector

clusterClaimSelector

Name	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

1.8.8. PlacementDecisions API (v1beta1)

1.8.8.1. 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**。

1.8.8.1.1. URI scheme

BasePath : /kubernetes/apis

Schemes : HTTPS

1.8.8.1.2. Tags

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

1.8.8.2. 路径

1.8.8.2.1. 查询所有 PlacementDecisions

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

1.8.8.2.1.1. 描述

查询您的 **PlacementDecisions** 获取更多详细信息。

1.8.8.2.1.2. 参数

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

1.8.8.2.1.3. 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

1.8.8.2.1.4. 使用

- **placementdecision/yaml**

1.8.8.2.1.5. Tags

- **cluster.open-cluster-management.io**

1.8.8.2.2. 创建 PlacementDecision

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

1.8.8.2.2.1. 描述

创建 **PlacementDecision**。

1.8.8.2.2.2. 参数

类型	Name	描述	模式
标头	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

1.8.8.2.2.3. 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
500	内部服务错误	无内容
503	服务不可用	无内容

1.8.8.2.2.4. 使用

- **placementdecision/yaml**

1.8.8.2.2.5. Tags

- **cluster.open-cluster-management.io**

1.8.8.2.2.6. HTTP 请求示例

1.8.8.2.2.6.1. 请求正文

```
{
  "apiVersion": "cluster.open-cluster-management.io/v1beta1",
  "kind": "PlacementDecision",
  "metadata": {
    "labels": {
      "cluster.open-cluster-management.io/placement": "placement1"
    }
  },
  "name": "placement1-decision1",
```

```

"namespace": "ns1"
},
"status" : {}
}

```

1.8.8.2.3. 查询单个 PlacementDecision

```

GET /cluster.open-cluster-
management.io/v1beta1/namespaces/{namespace}/placementdecisions/{placementdecision_n
ame}

```

1.8.8.2.3.1. 描述

查询单个 **PlacementDecision** 获取更多详细信息。

1.8.8.2.3.2. 参数

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

1.8.8.2.3.3. 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

1.8.8.2.3.4. Tags

- **cluster.open-cluster-management.io**

1.8.8.2.4. 删除 PlacementDecision

```
DELETE /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

1.8.8.2.4.1. 描述

删除单个 PlacementDecision。

1.8.8.2.4.2. 参数

类型	Name	描述	模式
标头	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

1.8.8.2.4.3. 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

1.8.8.2.4.4. Tags

- **cluster.open-cluster-management.io**

1.8.8.3. 定义

1.8.8.3.1. PlacementDecision

Name	描述	模式
apiVersion <i>必需</i>	版本化的 PlacementDecision schema	字符串
kind <i>必需</i>	代表 REST 资源的字符串值。	字符串
metadata <i>必需</i>	PlacementDecision 的元数据。	对象

1.9. 故障排除

在使用故障排除指南前，您可以运行 **oc adm must-gather** 命令来收集详情、日志和步骤来调试问题。如需了解更多详细信息，请参阅 [运行 **must-gather** 命令进行故障排除](#)。

另外，查看您基于角色的访问。详情请参阅 [multicluster engine operator 基于角色的访问控制](#)。

1.9.1. 记录的故障排除

查看 [multicluster engine operator 故障排除主题列表](#)：

安装：

要查看安装任务的主要文档，请参阅[安装和升级多集群引擎 operator](#)。

- [安装状态故障排除处于安装或待处理状态](#)
- [重新安装失败的故障排除](#)

集群管理：

要查看有关管理集群的主要文档，请参阅 [集群生命周期简介](#)。

- 将 **day-two** 节点添加到现有集群的故障排除会失败，并显示待处理用户操作
- 离线集群的故障排除
- 受管集群导入失败的故障排除
- 重新导入集群失败并显示未知颁发机构错误
- 对带有待处理导入状态的集群进行故障排除
- 证书更改后导入的集群离线故障排除
- 集群状态从离线变为可用的故障排除
- **VMware vSphere** 上创建集群的故障排除
- 集群在控制台中带有待处理或失败状态的故障排除
- **OpenShift Container Platform** 版本 **3.11** 集群导入失败的故障排除
- 带有降级条件的 **Klusterlet** 故障排除
- 删除集群后命名空间会保留
- 导入集群时出现自动 **Auto-import-secret-exists** 错误
- 在创建放置后缺少 **PlacementDecision** 进行故障排除

- 对 [Dell 硬件上的裸机主机](#) 发现失败的故障排除

1.9.2. 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

1.9.2.1. `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。

1.9.2.2. `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`。

注：如果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 --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。

1.9.2.3. 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。

- 运行 **must-gather** 命令进行故障排除

1.9.3. 故障排除：将 **day-two** 节点添加到现有集群失败，并显示待处理的用户操作

将节点或缩减到由 **multicluster engine** 为 **Kubernetes operator** 创建的现有集群中，且使用 **Zero Touch Provisioning** 或 **Host inventory create** 方法在安装过程中会失败。安装过程在发现阶段可以正常工作，但在安装阶段失败。

网络的配置失败。在集成控制台中的 **hub** 集群中看到 **Pending** 用户操作。在描述中，您可以看到它在重启步骤中失败。

有关失败的错误消息不准确，因为安装主机中运行的代理无法报告信息。

1.9.3.1. 症状：为第二天 **worker** 安装失败

发现阶段后，主机将重启以继续安装，但它无法配置网络。检查以下症状和信息：

- 在集成控制台中的 **hub** 集群中，检查添加节点上的 **Pending** 用户操作，并带有 **Rebooting** 指示符：

```
This host is pending user action. Host timed out when pulling ignition. Check the host console... Rebooting
```

- 在 **Red Hat OpenShift Container Platform** 配置受管集群中，检查现有集群的 **MachineConfig**。检查 **MachineConfig** 是否在以下目录中创建任何文件：

- **/sysroot/etc/NetworkManager/system-connections/**

- **/sysroot/etc/sysconfig/network-scripts/**

- 在安装主机的终端中，检查失败的主机是否有以下消息：您可以使用 **journalctl** 查看日志消息：

info: networking config is defined in the real root

info: will not attempt to propagate initramfs networking

如果您在日志中收到最后一条消息，则不会传播网络配置，因为它已在之前在 **Symptom** 中列出的文件夹中找到现有网络配置。

1.9.3.2. 解决问题：重新创建节点合并网络配置

执行以下任务以在安装过程中使用正确的网络配置：

1. 从 **hub** 集群中删除节点。
2. 重复前面的流程，以同样的方式安装节点。
3. 使用以下注解创建节点的 **BareMetalHost** 对象：

```
"bmac.agent-install.openshift.io/installer-args": ["--append-karg", "\"coreos.force_persist_ip\""]
```

节点开始安装。在发现阶段后，节点会在现有集群的更改和初始配置之间合并网络配置。

1.9.4. 安装状态故障排除处于安装或待处理状态

安装 **multicluster engine operator** 时，**MultiClusterEngine** 会一直处于 **Installing** 阶段，或者多个 **pod** 维护 **Pending** 状态。

1.9.4.1. 症状：一直处于 **Pending** 状态

安装 **MultiClusterEngine** 后超过十分钟，来自 **MultiClusterEngine** 资源的 **status.components** 字段的一个或多个组件报告 **ProgressDeadlineExceeded**。这可能是集群中的资源限制的问题。

检查安装了 **MultiClusterEngine** 的命名空间中的 **pod**。您可能会看到 **Pending** 状态，如下所示：

```
reason: Unschedulable
message: '0/6 nodes are available: 3 Insufficient cpu, 3 node(s) had taint {node-role.kubernetes.io/master:
```

```
}, that the pod didn't tolerate.'
```

在这种情况下，集群中 **worker** 节点资源不足以运行该产品。

1.9.4.2. 解决问题：调整 **worker** 节点大小

如果您有这个问题，则需要使用更大或多个 **worker** 节点来更新集群。如需了解[调整集群大小](#)的信息，请参阅[调整集群大小](#)。

1.9.5. 重新安装失败的故障排除

重新安装多集群引擎 **operator** 时，**pod** 不会启动。

1.9.5.1. 症状：重新安装失败

如果在安装 **multicluster engine operator** 后 **pod** 没有启动，通常是因为之前安装 **multicluster engine operator** 的项目在卸载时无法正确删除。

在本例中，**pod** 在完成安装过程后没有启动。

1.9.5.2. 解决问题：重新安装失败

如果您有这个问题，请完成以下步骤：

1. 按照 [卸载](#) 中的步骤，运行卸载过程来删除当前的组件。
2. 按照[安装 Helm](#) 中的内容，安装 **Helm CLI** 二进制版本 **3.2.0** 或更新版本。
3. 确保您的 **Red Hat OpenShift Container Platform CLI** 被配置为运行 **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](#)。
4. 将以下脚本复制到一个文件中：

```
#!/bin/bash
```

```
MCE_NAMESPACE=<namespace>
oc delete multiclusterengine --all
oc delete apiservice v1.admission.cluster.open-cluster-management.io
v1.admission.work.open-cluster-management.io
oc delete crd discoveredclusters.discovery.open-cluster-management.io
discoveryconfigs.discovery.open-cluster-management.io
oc delete mutatingwebhookconfiguration ocm-mutating-webhook
managedclustermutators.admission.cluster.open-cluster-management.io
oc delete validatingwebhookconfiguration ocm-validating-webhook
oc delete ns $MCE_NAMESPACE
```

将脚本中的 **<namespace >** 替换为安装 **multicluster engine operator** 的命名空间的名称。确保指定正确的命名空间，因为命名空间会被清理和删除。

5. 运行该脚本以删除以前安装中的内容。
6. 运行安装。请参阅[在线安装](#)。

1.9.6. 离线集群的故障排除

一些常见的原因会导致集群显示离线状态。

1.9.6.1. 症状：集群状态为离线

完成创建集群的步骤后，您无法从 **Red Hat Advanced Cluster Management** 控制台访问集群，集群的状态为离线 (**offline**)。

1.9.6.2. 解决问题：集群状态为离线

1. 确定受管集群是否可用。您可以在 **Red Hat Advanced Cluster Management** 控制台的 **Clusters** 区域中进行检查。

如果不可用，请尝试重启受管集群。

2. 如果受管集群状态仍处于离线状态，完成以下步骤：
 - a. 在 **hub** 集群上运行 **oc get managedcluster <cluster_name> -o yaml** 命令。将 **<cluster_name>** 替换为集群的名称。

- b. 找到 **status. conditionss** 部分。
- c. 检查 **type: ManagedClusterConditionAvailable** 信息并解决相关的问题。

1.9.7. 受管集群导入失败的故障排除

如果集群导入失败，您可以执行一些步骤来确定集群导入失败的原因。

1.9.7.1. 症状：导入的集群不可用

完成导入集群的步骤后，您无法从控制台访问它。

1.9.7.2. 解决问题：导入的集群不可用

在尝试导入集群后，有几个可能的原因会导致导入集群的不可用。如果集群导入失败，请完成以下步骤，直到找到失败导入的原因：

1. 在 **hub** 集群中，运行以下命令来确保导入控制器正在运行。

```
kubectl -n multicluster-engine get pods -l app=managedcluster-import-controller-v2
```

您应该会看到两个正在运行的 **pod**。如果任何一个 **pod** 没有运行，请运行以下命令来查看日志以确定原因：

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1
```

2. 在 **hub** 集群中，运行以下命令以确定受管集群导入 **secret** 是否由导入控制器成功生成：

```
kubectl -n <managed_cluster_name> get secrets <managed_cluster_name>-import
```

如果导入 **secret** 不存在，请运行以下命令来查看导入控制器的日志条目，并确定它没有被创建的原因：

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1 | grep importconfig-controller
```

3.

在 **hub** 集群中，如果您的受管集群是 **local-cluster**，或者由 **Hive** 置备，或者具有自动导入 **secret**，请运行以下命令来检查受管集群的导入状态。

```
kubectl get managedcluster <managed_cluster_name> -o=jsonpath='{range
.status.conditions[*]}.type{"\t"}{.status{"\t"}{.message{"\n"}}{end}' | grep
ManagedClusterImportSucceeded
```

如果 **ManagedClusterImportSucceeded** 条件为 **true**，则命令的结果表示失败的原因。

4.

检查受管集群的 **Klusterlet** 状态是否有降级条件。请参阅 [带有降级条件的 Klusterlet 故障排除](#)，以查找 **Klusterlet** 被降级的原因。

1.9.8. 重新导入集群失败并显示未知颁发机构错误

如果您在将受管集群重新导入到多集群引擎 **operator hub** 集群时遇到问题，请按照以下步骤排除此问题。

1.9.8.1. 症状：重新导入集群失败并显示未知颁发机构错误

使用 **multicluster engine operator** 置备 **OpenShift Container Platform** 集群后，当更改或将 **API** 服务器证书添加到 **OpenShift Container Platform** 集群时，重新导入集群可能会失败，并显示 **x509: certificate signed by unknown authority** 错误。

1.9.8.2. 鉴别问题：重新导入集群失败并显示未知颁发机构错误

在重新导入受管集群后，运行以下命令在 **multicluster engine operator hub** 集群上获取导入控制器日志：

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 -f
```

如果出现以下错误日志，受管集群 **API** 服务器证书可能会改变：

```
ERROR Reconciler error {"controller": "clusterdeployment-controller", "object":
{"name":"awscluster1","namespace":"awscluster1"}, "namespace": "awscluster1", "name":
"awscluster1", "reconcileID": "a2cccf24-2547-4e26-95fb-f258a6710d80", "error": "Get
\"https://api.awscluster1.dev04.red-chesterfield.com:6443/api?timeout=32s\": x509: certificate
signed by unknown authority"}
```

要确定受管集群 **API** 服务器证书是否已更改，请完成以下步骤：

1. 运行以下命令，将 **your-managed-cluster-name** 替换为受管集群的名称来指定受管集群名称：

```
cluster_name=<your-managed-cluster-name>
```

2. 运行以下命令获取受管集群 **kubeconfig secret** 名称：

```
kubeconfig_secret_name=$(oc -n ${cluster_name} get clusterdeployments ${cluster_name} -ojsonpath='{.spec.clusterMetadata.adminKubeconfigSecretRef.name}')
```

3. 运行以下命令，将 **kubeconfig** 导出到新文件：

```
oc -n ${cluster_name} get secret ${kubeconfig_secret_name} -ojsonpath={.data.kubeconfig} | base64 -d > kubeconfig.old
```

```
export KUBECONFIG=kubeconfig.old
```

4. 运行以下命令，使用 **kubeconfig** 从受管集群获取命名空间：

```
oc get ns
```

如果您收到类似以下消息的错误，您的集群 **API** 服务器符已更改，且 **kubeconfig** 文件无效。

无法连接到服务器：**x509: certificate signed by unknown authority**

1.9.8.3. 解决问题：重新导入集群失败并显示未知颁发机构错误

受管集群管理员必须为受管集群创建一个新的有效的 **kubeconfig** 文件。

创建新的 **kubeconfig** 后，执行以下步骤为受管集群更新新的 **kubeconfig**：

1. 运行以下命令来设置 **kubeconfig** 文件路径和集群名称。将 **<path_to_kubeconfig>** 替换为新 **kubeconfig** 文件的路径。将 **<managed_cluster_name>** 替换为受管集群的名称：

```
cluster_name=<managed_cluster_name>
kubeconfig_file=<path_to_kubeconfig>
```

2. 运行以下命令来对新的 **kubeconfig** 进行编码：

```
kubeconfig=$(cat ${kubeconfig_file} | base64 -w0)
```

注：在 **macOS** 中运行以下命令：

```
kubeconfig=$(cat ${kubeconfig_file} | base64)
```

3. 运行以下命令来定义 **kubeconfig json** 补丁：

```
kubeconfig_patch="[{"op":"replace", "path":"/data/kubeconfig",
"value":"${kubeconfig}"}, {"op":"replace", "path":"/data/raw-kubeconfig",
"value":"${kubeconfig}"}]"
```

4. 运行以下命令，从受管集群检索您的管理员 **kubeconfig secret** 名称：

```
kubeconfig_secret_name=$(oc -n ${cluster_name} get clusterdeployments ${cluster_name} -
ojsonpath='{.spec.clusterMetadata.adminKubeconfigSecretRef.name}')
```

5. 运行以下命令，使用您的新 **kubeconfig** 对管理员 **kubeconfig secret** 进行补丁：

```
oc -n ${cluster_name} patch secrets ${kubeconfig_secret_name} --type='json' -
p="${kubeconfig_patch}"
```

1.9.9. 对带有待处理导入状态的集群进行故障排除

如果在集群的控制台上持续接收到 **Pending import** (待处理到) 信息时，请按照以下步骤排除此问题。

1.9.9.1. 症状：集群处于待处理导入状态

在使用 **Red Hat Advanced Cluster Management** 控制台导入一个集群后，出现在控制台中的集群带有 **Pending import** 状态。

1.9.9.2. 鉴别问题：集群处于待处理导入状态

1.

在受管集群中运行以下命令查看有问题的 **Kubernetes pod** 的名称：

```
kubectl get pod -n open-cluster-management-agent | grep klusterlet-registration-agent
```

2.

在受管集群中运行以下命令查找错误的日志条目：

```
kubectl logs <registration_agent_pod> -n open-cluster-management-agent
```

把 **registration_agent_pod** 替换为在第 1 步中获得的 **pod** 名称。

3.

在返回的结果中搜索显示有网络连接问题的内容。示例包括：**no such host**。

1.9.9.3. 解决问题：集群处于待处理导入状态

1.

通过在 **hub** 集群中输入以下命令来检索有问题的端口号：

```
oc get infrastructure cluster -o yaml | grep apiServerURL
```

2.

确保来自受管集群的主机名可以被解析，并确保建立到主机和端口的出站连接。

如果无法通过受管集群建立通信，集群导入就不完整。受管集群的集群状态将会是 **Pending import**。

1.9.10. 证书更改后导入的集群离线故障排除

虽然支持安装自定义的 **apiserver** 证书，但在更改证书前导入的一个或多个集群会处于 **offline** 状态。

1.9.10.1. 症状：证书更改后集群处于离线状态

完成更新证书 **secret** 的步骤后，在线的一个或多个集群现在在控制台中显示 离线状态。

1.9.10.2. 鉴别问题：证书更改后集群处于离线状态

更新自定义 API 服务器证书信息后，在新证书前导入并运行的集群会处于 **offline** 状态。

表示证书有问题的错误会出现在离线受管集群的 **open-cluster-management-agent** 命名空间中的 **pod** 日志中。以下示例与日志中显示的错误类似：

work-agent 的日志：

```
E0917 03:04:05.874759    1 manifestwork_controller.go:179] Reconcile work test-1-klusterlet-
addon-workmgr fails with err: Failed to update work status with err Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks/test-
1-klusterlet-addon-workmgr": x509: certificate signed by unknown authority
E0917 03:04:05.874887    1 base_controller.go:231] "ManifestWorkAgent" controller failed to sync
"test-1-klusterlet-addon-workmgr", err: Failed to update work status with err Get "api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks/test-
1-klusterlet-addon-workmgr": x509: certificate signed by unknown authority
E0917 03:04:37.245859    1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1.ManifestWork: failed to list *v1.ManifestWork: Get "api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks?
resourceVersion=607424": x509: certificate signed by unknown authority
```

registration-agent 的日志：

```
I0917 02:27:41.525026    1 event.go:282] Event(v1.ObjectReference{Kind:"Namespace",
Namespace:"open-cluster-management-agent", Name:"open-cluster-management-agent", UID:"",
APIVersion:"v1", ResourceVersion:"", FieldPath:"}): type: 'Normal' reason:
'ManagedClusterAvailableConditionUpdated' update managed cluster "test-1" available condition to
"True", due to "Managed cluster is available"
E0917 02:58:26.315984    1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1beta1.CertificateSigningRequest: Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?
allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true": x509: certificate
signed by unknown authority
E0917 02:58:26.598343    1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1.ManagedCluster: Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?
allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true": x509: certificate
signed by unknown authority
E0917 02:58:27.613963    1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1.ManagedCluster: failed to list *v1.ManagedCluster: Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?
allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true": x509: certificate
signed by unknown authority
```

1.9.10.3. 解决问题： 证书更改后集群处于离线状态

如果您的受管集群是使用 **multicluster engine operator** 创建的 **local-cluster** 或受管集群，则必须等待 **10 分钟或更长时间**重新导入受管集群。

要立即重新导入受管集群，您可以删除 **hub** 集群上的受管集群导入 **secret**，并使用 **multicluster engine operator** 重新导入它。运行以下命令：

```
oc delete secret -n <cluster_name> <cluster_name>-import
```

将 **<cluster_name>** 替换为您要导入的受管集群的名称。

如果要重新导入使用 **multicluster engine operator** 导入的受管集群，请完成以下步骤再次导入受管集群：

1. 在 **hub** 集群中，运行以下命令来重新创建受管集群导入 **secret**：

```
oc delete secret -n <cluster_name> <cluster_name>-import
```

将 **<cluster_name>** 替换为您要导入的受管集群的名称。

2. 在 **hub** 集群中，运行以下命令来将受管集群导入 **secret** 公开给 **YAML** 文件：

```
oc get secret -n <cluster_name> <cluster_name>-import -ojsonpath='{.data.import\.yaml}' | base64 --decode > import.yaml
```

将 **<cluster_name>** 替换为您要导入的受管集群的名称。

3. 在受管集群中，运行以下命令来应用 **import.yaml** 文件：

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

注：前面的步骤不会从 **hub** 集群中分离受管集群。步骤使用受管集群中的当前设置更新所需的清单，包括新证书信息。

1.9.11. 集群状态从离线变为可用的故障排除

在没有对环境或集群进行任何手工更改的情况下，受管集群的状态在 **offline**（离线）和 **available**（可用）间转换。

1.9.11.1. 症状：集群状态从离线变为可用

当将受管集群连接到 **hub** 集群的网络不稳定时，**hub** 集群所报告的受管集群的状态在离线和可用之间不断转换。

1.9.11.2. 解决问题：集群状态从离线变为可用

要尝试解决这个问题，请完成以下步骤：

1.

输入以下命令在 **hub** 集群上编辑 **ManagedCluster** 规格：

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

将 **cluster-name** 替换为您的受管集群的名称。

2.

在 **ManagedCluster** 规格中增加 **leaseDurationSeconds** 的值。默认值为 **5** 分钟，但可能没有足够的时间来保持与网络问题的连接。为租期指定较长的时间。例如，您可以将这个值提高为 **20** 分钟。

1.9.12. VMware vSphere 上创建集群的故障排除

如果您在 **VMware vSphere** 上创建 **Red Hat OpenShift Container Platform** 集群时遇到问题，请查看以下故障排除信息以查看它们是否解决了您的问题。

注：当集群创建过程在 **VMware vSphere** 上失败时，您将无法使用该链接来查看日志。如果发生这种情况，您可以通过查看 **thehive-controllers pod** 的日志来找出问题。**hive-controllers** 日志位于 **hive** 命名空间中。

1.9.12.1. 受管集群创建失败并显示证书 IP SAN 错误

1.9.12.1.1. 症状：Managed 集群创建失败并显示证书 IP SAN 错误

在 **VMware vSphere** 上创建新的 **Red Hat OpenShift Container Platform** 集群后，集群会失败，并显示一个错误消息，显示证书 **IP SAN** 错误。

1.9.12.1.2. 鉴别问题：管理的集群创建失败并显示证书 IP SAN 错误

受管集群的部署失败，并在部署日志中返回以下错误：

```
time="2020-08-07T15:27:55Z" level=error msg="Error: error setting up new vSphere SOAP client:
Post https://147.1.1.1/sdk: x509: cannot validate certificate for xx.xx.xx.xx because it doesn't contain
any IP SANs"
time="2020-08-07T15:27:55Z" level=error
```

1.9.12.1.3. 解决问题：管理的集群创建失败，并显示证书 IP SAN 错误

使用 **VMware vCenter** 服务器完全限定主机名，而不是凭证中的 **IP** 地址。您还可以更新 **VMware vCenter CA** 证书以包含 **IP SAN**。

1.9.12.2. 受管集群创建失败并显示未知证书颁发机构

1.9.12.2.1. 症状：管理集群创建失败并显示未知证书颁发机构

在 **VMware vSphere** 上创建新的 **Red Hat OpenShift Container Platform** 集群后，集群会失败，因为证书由未知颁发机构签名。

1.9.12.2.2. 鉴别问题：Managed 集群创建失败并显示未知证书颁发机构

受管集群的部署失败，并在部署日志中返回以下错误：

```
Error: error setting up new vSphere SOAP client: Post https://vspherehost.com/sdk: x509: certificate
signed by unknown authority"
```

1.9.12.2.3. 解决问题：管理的集群创建失败并显示未知证书颁发机构

确保您在创建凭证时从证书认证机构输入了正确的证书。

1.9.12.3. 受管集群创建带有过期证书失败

1.9.12.3.1. 情况：集群创建失败并显示过期的证书

在 **VMware vSphere** 上创建新的 **Red Hat OpenShift Container Platform** 集群后，集群会失败，因为证书已过期或者无效。

1.9.12.3.2. 鉴别问题：管理的集群创建失败并显示过期的证书

受管集群的部署失败，并在部署日志中返回以下错误：

```
x509: certificate has expired or is not yet valid
```

1.9.12.3.3. 解决问题：管理的集群创建失败并显示过期的证书

确保同步了 **ESXi** 主机上的时间。

1.9.12.4. 受管集群创建失败且没有标记权限

1.9.12.4.1. 症状：管理集群创建失败且没有足够特权进行标记

在 **VMware vSphere** 上创建新的 **Red Hat OpenShift Container Platform** 集群后，集群会失败，因为没有足够的权限进行标记。

1.9.12.4.2. 鉴别问题：**Managed** 集群创建会失败，没有足够权限进行标记

受管集群的部署失败，并在部署日志中返回以下错误：

```
time="2020-08-07T19:41:58Z" level=debug msg="vsphere_tag_category.category: Creating..."
time="2020-08-07T19:41:58Z" level=error
time="2020-08-07T19:41:58Z" level=error msg="Error: could not create category: POST
https://vspherehost.com/rest/com/vmware/cis/tagging/category: 403 Forbidden"
time="2020-08-07T19:41:58Z" level=error
time="2020-08-07T19:41:58Z" level=error msg=" on ../tmp/openshift-install-436877649/main.tf line
54, in resource \"vsphere_tag_category\" \"category\":"
time="2020-08-07T19:41:58Z" level=error msg=" 54: resource \"vsphere_tag_category\" \"category\"
{"
```

1.9.12.4.3. 解决问题：管理的集群创建没有足够权限进行标记

确保 **VMware vCenter** 所需的帐户权限正确。如需更多信息，请参阅[删除的镜像 registry](#)。

1.9.12.5. 受管集群创建失败并显示无效的 dnsVIP

1.9.12.5.1. 症状：受管集群创建失败并显示无效的 dnsVIP

在 **VMware vSphere** 上创建新的 **Red Hat OpenShift Container Platform** 集群后，集群会失败，因为存在无效的 **dnsVIP**。

1.9.12.5.2. 鉴别问题：Managed 集群创建失败并显示无效的 dnsVIP

如果您在尝试使用 **VMware vSphere** 部署新受管集群时看到以下消息，这是因为您有一个较老的 **OpenShift Container Platform** 发行版本镜像，它不支持 **VMware Installer Provisioned Infrastructure (IPI)**：

```
failed to fetch Master Machines: failed to load asset \\\"Install Config\\\": invalid \\\"install-config.yaml\\\" file: platform.vsphere.dnsVIP: Invalid value: \\\"\\\": \\\"\\\" is not a valid IP
```

1.9.12.5.3. 解决问题：受管集群创建失败并显示无效的 dnsVIP

从支持 **VMware Installer Provisioned Infrastructure** 的 **OpenShift Container Platform** 版本中选择一个发行镜像。

1.9.12.6. 受管集群创建带有不正确的网络类型失败

1.9.12.6.1. 症状：集群创建失败并显示不正确的网络类型

在 **VMware vSphere** 上创建新的 **Red Hat OpenShift Container Platform** 集群后，集群会失败，因为指定的网络类型不正确。

1.9.12.6.2. 鉴别问题：管理的集群创建失败并显示不正确的网络类型

如果您在尝试使用 **VMware vSphere** 部署新受管集群时看到以下消息，这是因为您有一个旧的 **OpenShift Container Platform** 镜像，它不支持 **VMware Installer Provisioned Infrastructure (IPI)**：

```
time="2020-08-11T14:31:38-04:00" level=debug msg="vsphereprivate_import_ova.import: Creating..."
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=error msg="Error: rpc error: code = Unavailable desc = transport is closing"
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=fatal msg="failed to fetch Cluster: failed to generate asset \\\"Cluster\\\": failed to create cluster: failed to apply Terraform: failed to complete the change"
```

1.9.12.6.3. 解决问题：受管集群创建失败并显示不正确的网络类型

为指定的 **VMware** 集群选择一个有效的 **VMware vSphere** 网络类型。

1.9.12.7. 受管集群创建失败并显示磁盘更改错误

1.9.12.7.1. 症状：因为错误处理磁盘更改导致添加 VMware vSphere 受管集群失败

在 VMware vSphere 上创建新的 Red Hat OpenShift Container Platform 集群后，集群会失败，因为在处理磁盘更改时会出现错误。

1.9.12.7.2. 鉴别问题：添加 VMware vSphere 受管集群会因为处理磁盘更改出错而失败

日志中会显示类似以下内容的消息：

```
ERROR
ERROR Error: error reconfiguring virtual machine: error processing disk changes post-clone: disk.0:
ServerFaultCode: NoPermission: RESOURCE (vm-71:2000), ACTION (queryAssociatedProfile):
RESOURCE (vm-71), ACTION (PolicyIDByVirtualDisk)
```

1.9.12.7.3. 解决问题：因为错误处理磁盘更改导致 VMware vSphere 受管集群失败

使用 VMware vSphere 客户端为用户授予 *Profile-driven Storage Privileges* 的所有权限。

1.9.13. 集群在控制台中带有待处理或失败状态的故障排除

如果您在控制台中看到您创建的集群的状态为 *Pending* 或 *Failed*，请按照以下步骤排除问题。

1.9.13.1. 症状：集群在控制台中带有待处理或失败状态

在使用控制台创建新集群后，集群不会超过 *Pending* 状态或显示 *Failed* 状态。

1.9.13.2. 鉴别问题：集群在控制台中显示待处理或失败状态

如果集群显示 *Failed* 状态，进入集群的详情页面并使用提供的日志的链接。如果没有找到日志或集群显示 *Pending* 状态，请按照以下步骤检查日志：

-

流程 1

- 1.

在 **hub** 集群中运行以下命令，查看在命名空间中为新集群创建的 Kubernetes pod 的名称：

```
oc get pod -n <new_cluster_name>
```

使用您创建的集群名称替换 `new_cluster_name`。

2.

如果没有 `pod` 在列出的名称中包括 `provision` 字符串，则按照流程 2 继续进行。如果存在其标题中带有 `provision` 字符串的 `pod`，则在 `hub` 集群中运行以下命令以查看该 `pod` 的日志：

```
oc logs <new_cluster_name_provision_pod_name> -n <new_cluster_name> -c hive
```

将 `new_cluster_name_provision_pod_name` 替换为您创建的集群的名称，后接包含 `provision` 的 `pod` 名称。

3.

搜索日志中可能会解释造成问题的原因的错误信息。

•

流程 2

如果没有在其名称中带有 `provision` 的 `pod`，则代表问题在进程早期发生。完成以下步骤以查看日志：

1.

在 `hub` 集群中运行以下命令：

```
oc describe clusterdeployments -n <new_cluster_name>
```

使用您创建的集群名称替换 `new_cluster_name`。如需有关集群安装日志的更多信息，请参阅 [Red Hat OpenShift 文档中的 收集安装日志](#) 的内容。

2.

检查是否在资源的 `Status.Conditions.Message` 和 `Status.Conditions.Reason` 条目中存在有关此问题的额外信息。

1.9.13.3. 解决问题：集群在控制台中显示待处理或失败状态

在日志中找到错误后，确定如何在销毁集群并重新创建它之前解决相关的错误。

以下示例包括了一个选择不支持的区的日志错误，以及解决它所需的操作：

No subnets provided for zones

When you created your cluster, you selected one or more zones within a region that are not supported.在重新创建集群时完成以下操作之一以解决此问题：

- 在区域里（**region**）选择不同的区（**zone**）。
- 如果列出了其它区，则省略不支持的区。
- 为集群选择不同的区域。

在处理了日志中记录的问题后，销毁集群并重新创建它。

如需了解更多与创建集群 相关的信息，请参阅创建集群。

1.9.14. OpenShift Container Platform 版本 3.11 集群导入失败的故障排除

1.9.14.1. 症状：OpenShift Container Platform 版本 3.11 集群导入失败

试图导入 Red Hat OpenShift Container Platform 版本 3.11 集群后，导入会失败，并显示类似以下内容的日志消息：

```
customresourcedefinition.apiextensions.k8s.io/klusterlets.operator.open-cluster-management.io
configured
clusterrole.rbac.authorization.k8s.io/klusterlet configured
clusterrole.rbac.authorization.k8s.io/open-cluster-management:klusterlet-admin-aggregate-clusterrole
configured
clusterrolebinding.rbac.authorization.k8s.io/klusterlet configured
namespace/open-cluster-management-agent configured
secret/open-cluster-management-image-pull-credentials unchanged
serviceaccount/klusterlet configured
deployment.apps/klusterlet unchanged
klusterlet.operator.open-cluster-management.io/klusterlet configured
Error from server (BadRequest): error when creating "STDIN": Secret in version "v1" cannot be
handled as a Secret:
v1.Secret.ObjectMeta:
v1.ObjectMeta.TypeMeta: Kind: Data: decode base64: illegal base64 data at input byte 1313, error
found in #10 byte of ...|dhruy45="},"kind":"|... , bigger context
...|tye56u56u568yuo7i67i67i67o556574i"},"kind":"Secret","metadata":{"annotations":{"kube|...
```

1.9.14.2. 鉴别问题：OpenShift Container Platform 版本 3.11 集群导入失败

这通常是因为安装的 `kubectl` 命令行工具的版本为 **1.11** 或更早版本。运行以下命令，以查看您正在运行的 `kubectl` 命令行工具的版本：

```
kubectl version
```

如果返回的数据列出了 **1.11** 或更早版本，按照 [解决问题：OpenShift Container Platform 版本 3.11 集群导入失败](#) 中的内容进行解决。

1.9.14.3. 解决问题：OpenShift Container Platform 版本 3.11 集群导入失败

您可以通过完成以下步骤之一解决这个问题：

- 安装 `kubectl` 命令行工具的最新版本。
 1. 下载 `kubectl` 工具的最新版本：参阅 [Kubernetes 文档中的按照和设置 kubectl](#)。
 2. 升级 `kubectl` 工具后再次导入集群。
- 运行包含导入命令的文件。
 1. 启动 [使用 CLI 导入受管集群](#) 的步骤。
 2. 在创建用于导入集群的命令时，将该命令复制到名为 `import.yaml` 的 `YAML` 文件中。
 3. 运行以下命令从文件中再次导入集群：

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

1.9.15. 带有降级条件的 Klusterlet 故障排除

`Klusterlet` 降级条件可帮助诊断受管集群中的 `Klusterlet` 代理的状态。如果 `Klusterlet` 处于 `degraded` 条件，受管集群中的 `Klusterlet` 代理可能会出错，需要进行故障排除。对于设置为 `True` 的

Klusterlet 降级条件，请参见以下信息。

1.9.15.1. 症状：**Klusterlet** 处于降级状况

在受管集群中部署 **Klusterlet** 后，**KlusterletRegistrationDegraded** 或 **KlusterletWorkDegraded** 条件会显示 **True** 的状态。

1.9.15.2. 鉴别问题：**Klusterlet** 处于降级状况

1.

在受管集群中运行以下命令查看 **Klusterlet** 状态：

```
kubectl get klusterlets klusterlet -oyaml
```

2.

检查 **KlusterletRegistrationDegraded** 或 **KlusterletWorkDegraded** 以查看该条件是否被设置为 **True**。请根据 [解决这个问题](#) 的内容处理降级问题。

1.9.15.3. 解决问题：**Klusterlet** 处于降级状况

请查看以下降级状态列表，以及如何尝试解决这些问题：

- 如果 **KlusterletRegistrationDegraded** 条件的状态为 **True** 且状况原因为：**BootstrapSecretMissing**，您需要在 **open-cluster-management-agent** 命名空间中创建一个 **bootstrap secret**。
- 如果 **KlusterletRegistrationDegraded** 条件显示为 **True**，且状况原因为 **BootstrapSecretError** 或 **BootstrapSecretUnauthorized**，则当前的 **bootstrap secret** 无效。删除当前的 **bootstrap secret**，并在 **open-cluster-management-agent** 命名空间中重新创建有效的 **bootstrap secret**。
- 如果 **KlusterletRegistrationDegraded** 和 **KlusterletWorkDegraded** 显示为 **True**，且状况原因为 **HubKubeConfigSecretMissing**，请删除 **Klusterlet** 并重新创建它。
- 如果 **KlusterletRegistrationDegraded** 和 **KlusterletWorkDegraded** 显示为 **True**，则状况原因为：**ClusterNameMissing**、**KubeConfigMissing**、**HubConfigSecretError** 或 **HubConfigSecretUnauthorized**，从 **open-cluster-management-agent** 命名空间中删除 **hub** 集群 **kubeconfig secret**。注册代理将再次引导以获取新的 **hub** 集群 **kubeconfig secret**。
-

如果 **KlusterletRegistrationDegraded** 显示为 **True**，且状况原因为 **GetRegistrationDeploymentFailed** 或 **UnavailableRegistrationPod**，您可以检查条件消息以获取问题详情并尝试解决。

-

如果 **KlusterletWorkDegraded** 显示 **True**，且状况原因为 **GetWorkDeploymentFailed** 或 **UnavailableWorkPod**，您可以检查条件消息以获取问题详情并尝试解决。

1.9.16. 删除集群后命名空间会保留

当您删除受管集群时，该命名空间通常会作为移除集群过程的一部分被删除。在个别情况下，命名空间会和其中的一些工件一起被保留。在这种情况下，您必须手动删除命名空间。

1.9.16.1. 症状：删除集群后命名空间被保留

删除受管集群后，命名空间没有被删除。

1.9.16.2. 解决问题：删除集群后命名空间被保留

完成以下步骤以手动删除命名空间：

- 1.

运行以下命令以生成保留在 **<cluster_name>** 命名空间中的资源列表：

```
oc api-resources --verbs=list --namespaced -o name | grep -E
'^secrets|^serviceaccounts|^managedclusteraddons|^roles|^rolebindings|^manifestworks|^lease:|^managedclusterinfo|^appliedmanifestworks|^clusteroauths' | xargs -n 1 oc get --show-kind -
-ignore-not-found -n <cluster_name>
```

使用您要删除的集群的命名空间名称替换 **cluster_name**。

- 2.

输入以下命令来编辑列表，删除列表中状态不是 **Delete** 的资源：

```
oc edit <resource_kind> <resource_name> -n <namespace>
```

将 **resource_kind** 替换为资源类型。将 **resource_name** 替换为资源的名称。使用资源的命名空间的名称替换 **namespace**。

3. 在元数据中找到 **finalizer** 属性。
4. 使用 **vi** 编辑器的 **dd** 命令删除非 **Kubernetes finalizer**。
5. 输入 **:wq** 命令保存列表并退出 **vi** 编辑器。
6. 输入以下命令来删除命名空间：

```
oc delete ns <cluster-name>
```

使用您要删除的命名空间的名称替换 **cluster-name**。

1.9.17. 导入集群时出现自动 **Auto-import-secret-exists** 错误

集群导入失败，并显示出错信息：**auto import secret exists**。

1.9.17.1. 症状：导入集群时出现 **Auto import secret exists** 错误

当导入 **hive** 集群以进行管理时，会显示 **auto-import-secret already exists** 错误。

1.9.17.2. 解决问题：导入集群时出现 **Auto import secret exists** 错误

当您试图导入之前管理的集群时，会出现此问题。如果出现这种情况，当您尝试重新导入集群时，**secret** 会发生冲突。

要临时解决这个问题，请完成以下步骤：

1. 在 **hub** 集群中运行以下命令来手工删除存在的 **auto-import-secret**：

```
oc delete secret auto-import-secret -n <cluster-namespace>
```

将 **cluster-namespace** 替换为集群的命名空间。

2.

按照 [将目标受管集群导入到 hub 集群的步骤](#) 再次导入集群。

集群已导入。

1.9.18. 在创建放置后缺少 **PlacementDecision** 进行故障排除

如果在创建放置后没有生成 **PlacementDecision**，请按照以下步骤排除此问题。

1.9.18.1. 症状：创建放置后缺少 **PlacementDecision**

创建放置后，不会自动生成一个 **PlacementDecision**。

1.9.18.2. 解决问题：在创建放置后缺少 **PlacementDecision**

要解决这个问题，请完成以下步骤：

1.

运行以下命令检查放置条件：

```
kubectl describe placement <placement-name>
```

将 **placement-name** 替换为放置的名称。

输出可能类似以下示例：

```
Name:      demo-placement
Namespace: default
Labels:    <none>
Annotations: <none>
API Version: cluster.open-cluster-management.io/v1beta1
Kind:      Placement
Status:
  Conditions:
    Last Transition Time:  2022-09-30T07:39:45Z
    Message:              Placement configurations check pass
    Reason:                Succeedconfigured
    Status:                False
    Type:                  PlacementMisconfigured
    Last Transition Time:  2022-09-30T07:39:45Z
    Message:              No valid ManagedClusterSetBindings found in placement
```

```

namespace
  Reason:          NoManagedClusterSetBindings
  Status:          False
  Type:            PlacementSatisfied
  Number Of Selected Clusters: 0

```

2.

在输出中检查 **PlacementMisconfigured** 和 **PlacementSatisfied** 的 **Status** :

- 如果 **PlacementMisconfigured Status** 为 **true**，则您的 **Placement** 具有配置错误。检查包含的消息，了解配置错误的详情以及如何解决它们。
- 如果 **PlacementSatisfied Status** 为 **false**，则没有受管集群满足您的放置。检查包含的消息以获取更多详细信息以及如何解决错误。在上例中，放置命名空间中没有找到 **ManagedClusterSetBindings**。

3.

您可以检查 **Events** 中每个集群的分数，以了解某些具有较低分数的集群没有被选择。输出可能类似以下示例：

```

Name:      demo-placement
Namespace: default
Labels:    <none>
Annotations: <none>
API Version: cluster.open-cluster-management.io/v1beta1
Kind:      Placement
Events:
  Type    Reason      Age    From          Message
  ----    -
  Normal  DecisionCreate 2m10s placementController Decision demo-placement-decision-1 is created with placement demo-placement in namespace default
  Normal  DecisionUpdate 2m10s placementController Decision demo-placement-decision-1 is updated with placement demo-placement in namespace default
  Normal  ScoreUpdate   2m10s placementController cluster1:0 cluster2:100 cluster3:200
  Normal  DecisionUpdate 3s    placementController Decision demo-placement-decision-1 is updated with placement demo-placement in namespace default
  Normal  ScoreUpdate   3s    placementController cluster1:200 cluster2:145 cluster3:189 cluster4:200

```

注：放置控制器分配一个分数，并为每个过滤的 **ManagedCluster** 生成事件。当集群分数更改时，放置控制器会生成新事件。

1.9.19. 对 Dell 硬件上的裸机主机发现失败的故障排除

如果在 **Dell** 硬件上发现裸机主机失败，则集成的 **Dell Remote Access Controller (iDRAC)** 可能会被配置为不允许来自未知证书颁发机构的证书。

1.9.19.1. 症状：在 Dell 硬件中发现裸机主机失败

完成使用基板管理控制器发现裸机主机的步骤后，会显示类似如下的错误消息：

```
ProvisioningError 51s metal3-baremetal-controller Image provisioning failed: Deploy step
deploy.deploy failed with BadRequestError: HTTP POST
https://<bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/VirtualMedia.
InsertMedia returned code 400. Base.1.8.GeneralError: A general error has occurred. See
ExtendedInfo for more information Extended information: [
{"Message": "Unable to mount remote share https://<ironic_address>/redfish/boot-<uuid>.iso.",
'MessageArgs': ["https://<ironic_address>/redfish/boot-<uuid>.iso"], "MessageArgs@odata.count": 1,
"MessageId": "IDRAC.2.5.RAC0720", "RelatedProperties": ["#/Image"],
"RelatedProperties@odata.count": 1, "Resolution": "Retry the operation.", "Severity": "Informational"}
]
```

1.9.19.2. 解决问题：在 Dell 硬件中发现裸机主机的故障

iDRAC 配置为不接受来自未知证书颁发机构的证书。

要绕过这个问题，请完成以下步骤禁用主机 **iDRAC** 基板管理控制器上的证书验证：

1. 在 **iDRAC** 控制台中，进入到 **Configuration > Virtual media > Remote file share**。
2. 将 **Expired** 或无效证书操作的值更改为 **Yes**。